UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Advanced Policy Optimization Algorithms with Flexible Trust Region Constraint

By ***Haotian Xu***

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of *Jie Lu*

University of Technology Sydney
Faculty of Engineering and Information Technology

June 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Haotian Xu*, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Computer Science*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note:
Signature removed prior to publication.

DATE: June, 2024

PLACE: Sydney, Australia

ABSTRACT

In reinforcement learning, trust region constraint aims to control the difference between two adjacent policies to be less than a pre-defined threshold, to stabilize the policy optimization. The landmark algorithm in this research field is trust region policy optimization (TRPO). However, how and where to apply this trust region control and its related techniques remains a challenging issue. In this thesis, we will address this problem from four perspectives.

To relax the trust region for promoting exploration ability, we propose trust region policy optimization via entropy regularization for KL divergence constraint. Its novelty lies in that a Shannon entropy regularization term is added to the constraint to adjust the difference between two consecutive policies directly to select some potentially problematic policies, rather than modifying the objective function as seen in existing literature.

To bound the trust region from below, we present twin trust region policy optimization. At first, we propose a reciprocal trust region policy optimization through exchanging the objective and the KL divergence constraint according to the reciprocal optimization technique, which induces a lower bound of step size search. Then TRPO and its reciprocal version are integrated to build twin TRPO, which has a lower bound and an upper bound for step size search, to facilitate the optimization procedure of TRPO.

To characterize the trust region better for different environments, we construct an enhanced proximal policy optimization (PPO) formulated as an unconstrained minimization. The primary PPO with a single tunable factor is revisited, which is converted into a linear combination to control the trade-off between the return and the KL divergence, and then is inserted into a parameterized alpha divergence to replace KL divergence to adjust the trust region constraint. This algorithm can be solved using gradient-based optimization technique.

To adjust the trust region and enhance the sample efficiency, we implement diversity-driven model ensemble trust region policy optimization. Firstly, we design a deep residual attention U-net with significantly fewer weights as our base models and add normalized Hilbert-Schmidt independence criterion as a regularization term to pursue model diversity explicitly for the model ensemble. Moreover, we propose an adaptive TRPO, where the parametric Renyi alpha divergence substitutes for KL divergence for measuring the trust region with the alpha value adaptively adjusted during training iterations.

Finally, the effectiveness and efficiency of these proposed reinforcement learning algorithms have been validated experimentally on several widely used benchmark environments.

Dedication

This thesis is dedicated to my beloved parents, whose unwavering support, encouragement, and sacrifices have shaped my journey. Your belief in me has been my greatest motivation. I am forever grateful for your guidance and inspiration, which have helped me reach this milestone. I also dedicate this thesis to my late maternal grandfather, whose wisdom and strength always inspire me since childhood

Acknowledgments

As I reflect on my journey as a Ph.D. candidate, I realize it has been a transformative experience filled with challenges and growth. Especially, the COVID-19 pandemic, which began at the end of 2019, brought a unique set of challenges. This process has not only been an academic exploration but also a valuable opportunity to learn from and connect with many remarkable individuals. None of this would have been possible without their support and guidance.

I would like to begin by expressing my heartfelt gratitude to my principal supervisor, Distinguished Professor Jie Lu. Her exceptional supervision, unwavering support, and insightful feedback have been instrumental in shaping my research and the development of this thesis. Her patience, expertise, and dedication have profoundly influenced my academic journey. I am also sincerely thankful to my co-supervisors, Professor Guangquan Zhang, Dr. Junyu Xuan, and Dr. Yan Zheng. Their mentorship, encouragement, and constructive criticism have greatly enriched my academic experience and encouraged me to think critically and explore my research from diverse perspectives.

As a member of the Australian Artificial Intelligence Institute (AAII), I would like to express my gratitude to all the staff and students of AAII for their support in our daily endeavors. A special thanks goes to Dr. Junyu Xuan and Dr. Tianyu Liu for his invaluable academic support and for sharing their knowledge, which has significantly impacted my work. I am also grateful to Dr. Bin Wang and Dr. Hang Yu for their encouragement and positivity, which kept me motivated during challenging times. I extend my appreciation to Dr. Hua Zuo and Dr. Yi Zhang for their assistance during my doctoral candidate assessment. Their insights and support played a crucial role in my progress and development.

Lastly, I want to express my deep gratitude to my family. Your constant support and encouragement have been my anchor throughout this journey. I am especially grateful for my parents,Äô support during the challenging times brought on by COVID-19. They have allowed me to pursue my dreams without boundaries. Thank you to my parents and all the other members of my family. . Thank you all for being an essential part of this transformative experience.

List of Publications

1. **Haotian Xu**, Zheng Yan, Junyu Xuan, Guangquan Zhang, Jie Lu. (2023). Improving proximal policy optimization with alpha divergence. Neurocomputing, 534, 94-105.

2. **Haotian Xu**, Junyu Xuan, Guangquan Zhang, Jie Lu. (2024). Trust region policy optimization via entropy regularization for Kullback-Leibler divergence constraint. Neurocomputing, 589, Article-127716 (12 pages).

3. **Haotian Xu**, Junyu Xuan, Guangquan Zhang, Jie Lu. (2024). Reciprocal trust region policy optimization. The 16th FLINS Conference on Computational Intelligence in Decision and Control & the 19th ISKE Conference on Intelligence Systems and Knowledge Engineering, Madrid, Spain, July 16-21, pp.187-194, 2024.

4. **Haotian Xu**, Zheng Yan, Junyu Xuan, Guangquan Zhang, Jie Lu. (2023). Diversity-driven model ensemble adaptive trust region policy optimization. Submitted to IEEE Transactions on Systems, Man, and Cybernetics: Systems (the first revision).

5. **Haotian Xu**, Junyu Xuan, Guangquan Zhang, Jie Lu. (2024). Twin trust region policy optimization. Submitted to IEEE Transactions on Systems, Man, and Cybernetics: Systems (the first revision).

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE **Page**

List of Tables

NOMENCLATURE

A. Abbreviation

A1. Overall

ADAM:	Adaptive Moment Estimation
AI:	Artificial Intelligence
Att:	Attention
CCA:	Canonical component anlysis
AUC:	Area under the curve
CEM:	Cross-entropy method
CKA:	Central kernel alignment
DNN:	Deep neural network
EL:	Ensemble learning
ER:	Entropy regularization
FC:	fully-connected
HSIC:	Hilbert-Schmidt independence criterion
KL:	Kullback-Leibler divergence
MBRL:	Model-base reinforcement learning
MDP:	Markov decision process
ME:	Model ensemble
MERL:	Model ensemble reinforcement learning
MFRL:	Model-free reinforcement learning
ML:	Machine learning
MLP:	Multi-layer perceptron
MMD:	Maximal mean discrepancy
MPC:	Model predictive control
NHSIC:	Normalized HSIC
NN:	Neural network
PCL:	Path consistency learning
PO:	policy optimization
RauNet:	Residual attention U-net
ReLU:	Rectified linear unit
Res:	Residual
RL:	Reinforcement learning
ROC:	Receiver operating characteristic

RS: Random shooting
TV: Total variance divergence
TR: Trust region
WRT: With respect to

A2. RL Methods

AC: Actor-critic
alphaPPO: Proximal policy optimization with alpha divergence
DQN: Deep Q network
En-TRPO: Entropy regularized Trust region policy optimization
ERC-TRPO: Trust region policy optimization via entropy regularized constraint
ERO-TRPO: Trust region policy optimization via entropy regularized objective
PPO: Proximal policy optimization
rTRPO: Reciprocal trust region policy optimization
SAC: Soft actor-critic
TRPO: Trust region policy optimization
TwinTRPO: Twin trust region policy optimization

A3. MBRL Methods

DA-TRPO: Diversity-driven model ensemble adaptive trust region policy optimization
MBPO: Model-based policy optimization
MB-MF: Model-based deep reinforcement learning with model-free fine-tuning
ME-TRPO: Model ensemble TRPO
PETS: Probabilistic ensemble with trajectory sampling
SLBO: Stochastic lower bound optimization

B. Symbols

$\mathscr{A}$:	Action space
$\mathscr{S}$:	State space
a_t :	An action in $\mathscr{A}$
s_t :	A state in $\mathscr{S}$
r_t :	A reward from the environment
γ	The discounted factor
$p(s'\|s_t,a_t)$:	A probability of the state transition from s_t to s' via executing an action a_t
$Q(s_t,a_t)$:	State-action value function
$V(s_t)$:	State value function
$A(s_t,a_t)$:	Advantage function
θ :	A parameter vector of the actor
$\pi_\theta(a_t\|s_t)$:	A policy using a parameter vector θ
$D_{KL}(p\|\|q)$:	KL divergence between two distributions p and q
$D_{AD}(p\|\|q)$:	alpha divergence between p and q
$D_{RA}(p\|\|q)$:	Renyi alpha divergence between p and q
H :	A Hessian matrix of the divergence wrt θ
g :	A gradient vector of the objective wrt θ
h :	A gradient vector of the entropy wrt θ

CHAPTER 1

INTRODUCTION

1.1 Background

Machine learning (ML) [16, 118, 139] is one of the hottest research areas in artificial intelligence (AI) [87, 104, 110], which covers three main learning paradigms: supervised learning, unsupervised learning and reinforcement learning [6, 10, 105].

Supervised learning typically involves learning a function that maps an input to an output based on sample input-output pairs. It uses labeled training data and a collection of training examples to infer a function. The most common supervised tasks are classification which separates the data, and regression which fits the data [10, 105].

Unsupervised learning analyzes unlabeled data sets, to extract generative features, identify meaningful trends and structures, and provide a physical exploratory. The most common unsupervised learning tasks are clustering, density estimation, association rule mining, anomaly detection and so on [10, 105].

Being different from the aforementioned two learning paradigms, reinforcement learning (RL) [34, 79, 81, 91, 93, 115, 123, 134, 151] tackles the unique problem of how to act optimally in an unknown dynamic environment, which is also referred to as data-driven sequential decision problem. The agent executes a sequence of actions in the environment, and, by doing so, manipulates the state of that environment. Through subsequent iterations of actions and feedback in the form of a reward, the agent learns how to act more optimally, as shown in Figure 1.1.

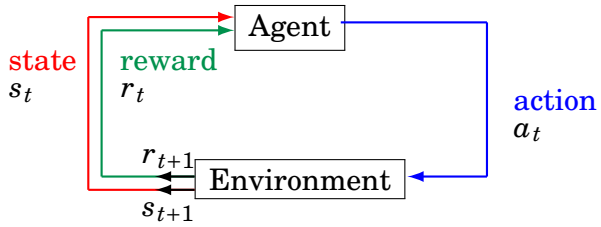


Figure 1.1: Reinforcement learning framework.

Considering the breakthrough of AlphaGO and AlphaZero [112, 113], reinforcement learning, especially deep reinforcement learning combining deep learning with reinforcement learning, has become a recognized technique for solving sequential decision making problems. This has also been extended and applied to various classical and novel real-world applications, like recommendation systems [1, 66], autonomous driving [57, 158], combinatorial optimization [31, 75], medical imaging [51, 162], health care [5, 152], game playing [116, 149] and generative AI [41, 142], and to list just a few.

Essentially, reinforcement learning aims to learn an optimal behavior policy by trial and error in an interactive environment. A policy defines how the learning agent will behave at a given time. Formally, a policy represents a mapping from a state to the probability of selecting a possible action [123]. The agent's goal is to improve this policy to obtain higher rewards in a process called policy optimization [34].

Policy optimization gives rise to two main kinds of reinforcement learning methods: value-based and policy-based optimizations [34], also referred to as value iteration and policy iteration. The former optimizes the action-state value function to produce the probability of an action being selected. Notable examples include Q-learning [138] and its deep version (DQN) [80] are two representative examples. These methods usually have lower variance in the estimates of expected discounted rewards and are generally implemented using an actor architecture [45]. In contrast, they are mostly suited to discrete action environments. Policy-based optimization optimizes the policy directly according to the sampled rewards. REINFORCE [141] is an example of this type of policy optimization. This category of methods generally suffers from a high variance in the estimates of the gradient, leading to a slow training procedure, and is implemented using a critic architecture [45].

Combining two strategies results in the more widely-used actor-critic group of techniques, which optimize the value function as guidance for policy improvements [123]. These methods usually consist of two steps: policy evaluation and policy improvement.

Widely-known methods include the actor-critic (AC) algorithm [60] and its variants [123] plus the trust region policy optimization (TRPO) algorithm [106] and its variant: proximal policy optimization (PPO) [107].

These actor and actor-critic methods originally aim to maximize the total expected discounted rewards, as described by the actor-critic algorithm [60]. Alternatively, an advantage function can be defined based on the state and the state-action value function, which can then be used to estimate a surrogate objective [106] based on importance sampling [36, 128]. In TRPO [106], this surrogate objective is maximized, when Kullback-Leibler (KL) divergence [7, 156] between two adjacent policies falls below a pre-determined threshold. This is a highly scalable and efficient approach to TRPO and it has been implemented in many successful applications.

In RL community, it is widely recognized that the trust region constraint is one of the most vital strategies to boost the success of TRPO. However, how and where to apply the trust region and its related techniques remains a challenging issue. In this thesis, we will investigate the following four aspects to address this problem.

For some application environments, such a strict constraint in TRPO possibly excludes all suspicious gradients. This typically stops the agent from fully exploring the environment and negatively impacts the performance [64, 71]. One way to encourage exploration is to insert Shannon entropy [29] or one of its related variants (e.g., log probability [71]) into the expected reward or surrogate objectives. For example, in the TRPO variant [103], entropy regularization is added to the surrogate objective. Similarly, Ma [71] integrated entropy into the reward, and log probability into the state value function and state-action value functions, to improve both TRPO and PPO. To the best of our knowledge, this regularization strategy has only been applied to the objectives of reinforcement learning algorithms. In this thesis, we argue for adding the regularization term to the constraint, to relax the trust region for promoting the exploration ability.

On the one hand, the KL divergence constraint in TRPO is essentially to bound the trust region from above, which prompts us whether to bound the trust region from below. On the other hand, the TRPO is solved approximately and iteratively, which depends on two factors: the search direction and the step size from the optimization view [15, 120]. The current solution can provide an upper bound of step size search, but no lower bound is considered. This situation has a dramatic impact on evaluating an optimal step size. In this thesis, we focus on how to find out a lower bound of step size search, to facilitate TRPO solution further.

The approximate solution to TRPO essentially is a second-order optimization tech-

nique [15, 120], which needs a Hessian matrix at each iteration. The PPO is a representative variant of TRPO, which builds an unconstrained optimization problem from TRPO using the Lagrangian technique. Nowadays, there are three detailed forms: primary, adaptive, and clipping. The clipping form is widely investigated experimentally, which applies a clipping function to measure the trust region indirectly. In our thesis, we argue for how to enhance the primary formulation effectively and simultaneously adjust the explicit trust region measure via alpha divergence for different environments.

The aforementioned reinforcement learning algorithms belong to model-free techniques, where the model indicates the dynamics of the environment. A main drawback of these techniques is that they require a lot of interaction with the environment. To improve the sample efficiency, model-based reinforcement learning is designed [69, 81, 92]. It is able to obtain millions of trajectories from the estimated model instead of expensive interactions with the real environment. A more accurate environment model is based on ensemble learning techniques [77, 99]. Further, TRPO is used as a basic policy optimization algorithm [63, 70]. Additionally, each model is mainly built by simple deep neural networks (i.e., multi-layer perceptron with ReLu activation function), which induces millions of weights to be optimized. In this thesis, we focus on how to reduce the number of weights via constructing a concise neural network architecture for the dynamics model to boost the sample efficiency, and how to enhance TRPO via Renyi alpha divergence to adjust the trust region automatically.

This research background is associated with trust region control, which will become our focus in this thesis.

1.2 Research Objectives and Outcomes

According to the aforementioned background, we list our research questions and their research outcomes and contributions in this section.

1.2.1 Research Questions

Concentrating on flexible trust region constraint, this study consists of four key Research Questions (RQ) listed as follows:

RQ 1: How to regularize trust region policy optimization, to relax its trust region for enhancing its exploration ability for policy optimization?

RQ 2: How to bound the trust region from below and to find out a lower bound of step size search, to facilitate the solution to trust region policy optimization?

RQ 3: How to characterize the trust region for different environments, to improve the performance for primary proximal policy optimization effectively?

RQ 4: How to construct a diversity-driven deep neural network architecture for dynamics model to boost the sample efficiency, and how to adaptively adjust the trust region for policy optimization?

1.2.2 Research Objectives

In order to answer these research questions, the corresponding Research Objectives (RO) are summarized as follows:

RO 1: To design a novel trust region policy optimization via regularizing K-L divergence constraint, thereby relaxing the trust region to enhance the exploration ability for policy optimization.

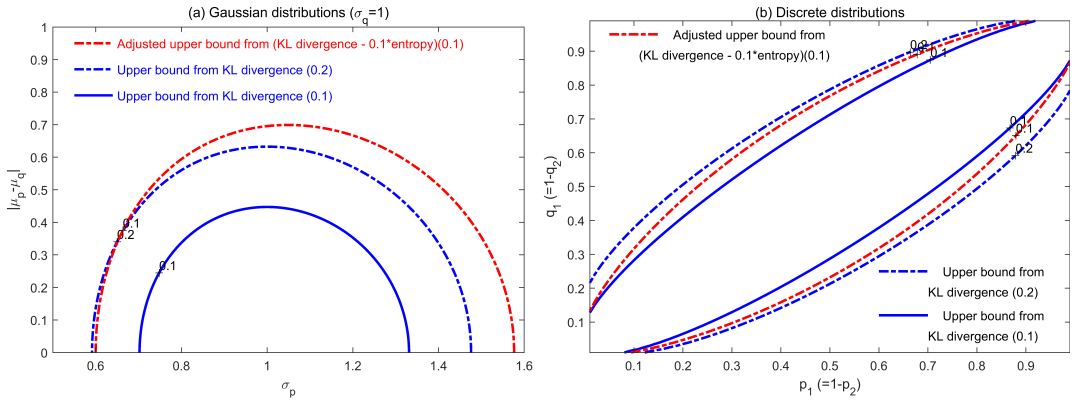


Figure 1.2: To relax the trust region from above, for Gaussian and discrete distributions.

The original trust region policy optimization is criticized for limiting its exploration ability due to its strict KL divergence constraint. To this end, a widely-used way is to embed an entropy regularization term to its objective. In our study, we alternatively add such a regularization to its constraint, resulting in a novel trust region policy optimization via entropy regularization for KL divergence constraint, which provides a new regularization way to enhance the exploration for trust region policy optimization, and relaxes the trust region from above in essence, as shown in Figure 1.2. In particular, the adjusted trust region covers a larger variance area in Figure 1.2(a) for two Gaussian distributions.

RO 2: To propose a twin trust region policy optimization to bound the trust region from below and thus to detect a lower bound of trust region, to facilitate the solution to trust region policy optimization, with a proper range of step size search.

The primary trust region policy optimization essentially bounds the trust region from above, so it is necessary to bound it from below. On the other hand, from optimization viewpoint, the nonlinear trust region policy optimization is solved via a linear approximation to its objective and a quadratic approximation to its KL divergence constraint. This solution depends on two aspects: a search direction and a step size. An optimal step size is associated with a proper step size interval. Now an upper bound of step size is provided, but no lower bound is considered yet. In this study, we leverage reciprocal optimization to exchange the objective and the constraint, to construct a reciprocal trust region policy optimization, which bounds the trust region from below, and whose approximate solution can give a lower bound of step size. Combining such two trust region policy optimization methods is to build a twin trust region policy optimization, which bounds the trust region from below and from above, and whose approximate solution has a proper interval for step size search. As shown in Figure 1.3, an adjusted lower bound of the trust region is automatically detected via the least return setting.

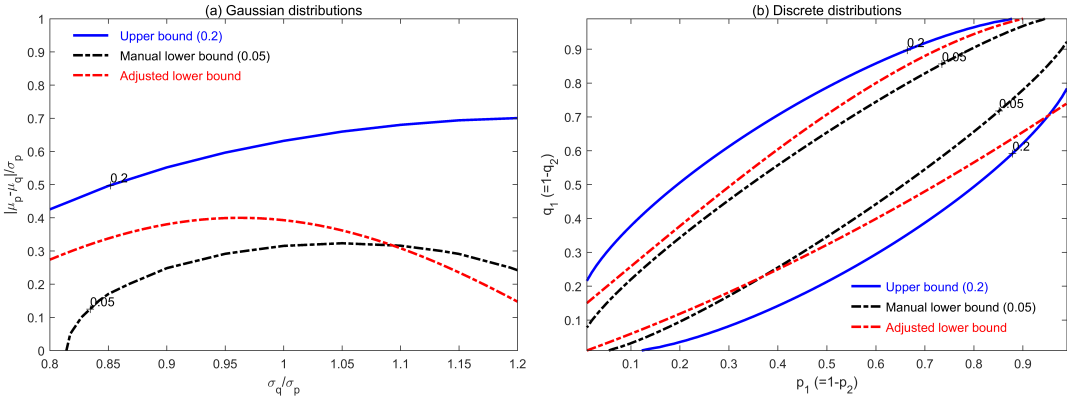


Figure 1.3: To detect the trust region from below, for Gaussian and discrete distributions.

RO 3: To construct an alpha proximal policy optimization via a parametric alpha divergence and a primary combined proximal policy optimization, to detect the trust region with a proper alpha divergence for each environment.

The above trust region policy optimization methods principally apply a second-order solution technique, which needs a Hessian matrix besides a gradient vector. Via the Lagrangian multiplier technique, this trust region policy optimization is converted into

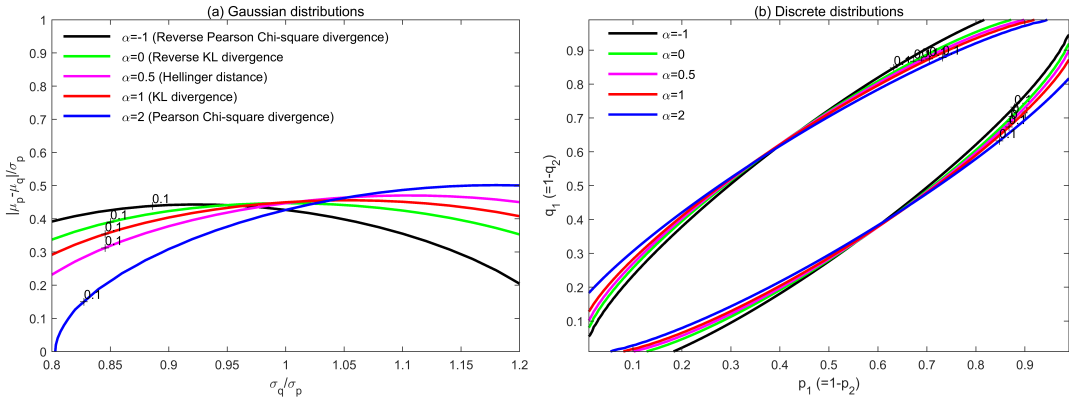


Figure 1.4: To show the different trust regions with different alpha values in alpha divergence.

an unconstrained problem, named proximal policy optimization, which has three forms: primary, adaptive, and clipping. The last clipping form is widely used in experimental comparison, but it utilizes a clipping function to measure the trust region indirectly. In this study, we revisit the primary form and then enhance its performance via a linear combination to control the trade-off between the surrogate return and the divergence, and alpha divergence to measure the trust region more effectively. This study also aims to apply alpha divergence to adjust the trust region for different environments. Figure 1.4 shows the distinct trust regions with different alpha values, which are detected by different environments, as shown in Figure 1.5.

RO 4: To build a diversity-driven model ensemble adaptive trust region policy optimization, to improve the sample efficiency and adjust the trust region adaptively.

The trust region policy optimization essentially belongs to an on-policy reinforcement learning technique, which executes numerous interactions with the real environment continuously and thus has a relatively lower sample efficiency. Model-based reinforcement learning builds a dynamics environment model using a small number of interactions with the true environment, and its ensemble form further reduces the model variance, to improve the sample efficiency. Nowadays the dynamics model is widely designed as a multi-layer perceptron with ReLU activation function, with millions of weights to be learned. In this thesis, we design a residual attention U-net for each dynamic model with significantly fewer weights and further use Hilbert-Schmidt independence criterion to enhance the diversity of learned models. For trust region policy optimization trained by interactions with the learned ensemble model, we replace the KL divergence with

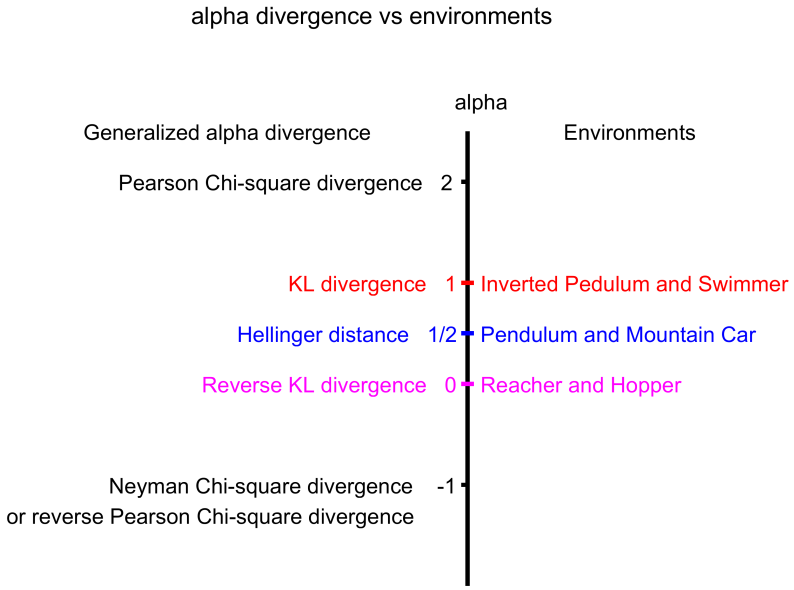


Figure 1.5: To detect different alpha values for different environments.

the Renyi alpha divergence and adaptively adjust the alpha value during the policy optimization. Our novel diversity driven model ensemble trust region policy optimization improves the sample efficiency and adjusts the trust region adaptively. Figure 1.6 illustrates some different trust regions created by different alpha values in Renyi alpha divergence, which would fit different return situations in the training stage.

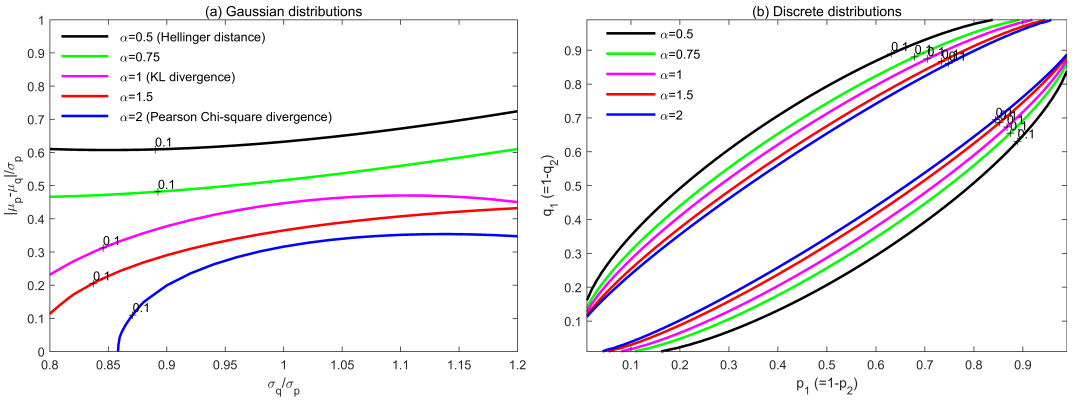


Figure 1.6: To adjust the upper bounds of trust region with different alpha values in Renyi alpha divergence.

1.2.3 Research Contributions

The innovation of this study can be concluded as following aspects:

1. To enhance the exploration ability for trust region policy optimization, we propose a novel entropy regularization strategy, which is to add an entropy regularization to the constraint, rather than to the (surrogate) objective widely used in those existing entropy regularization. Our strategy relaxes the trust region from above to enhance the exploration ability.

2. To bound the trust region from below and build a more effective solution to trust region policy optimization, we design a twin trust region policy optimization. At first, to bound the trust region from below, we introduce a reciprocal trust region policy optimization via exchanging the objective and the KL divergence constraint, according to reciprocal optimization. It induces a lower bound of step size search. Our twin trust region policy optimization is integrated by such two approaches, whose approximate solution has an upper bound from the original trust region policy optimization and a lower bound from reciprocal trust region policy optimization, to facilitate the policy optimization effectively.

3. To describe the trust region much better for different environments, On the proximal policy optimization, i.e., the most famous variant of trust region policy optimization, we revisit its primary proximal policy optimization and improve its performance further, via defining its linear combination formulation for tuning the balance of two terms conveniently and measuring the trust region with the parametric alpha divergence.

4. To improve the sample efficiency for trust region policy optimization, we design a residual attention U-net with fewer weights to construct the dynamics environment models and add a Hilbert-Schmidt independence criterion to increase the ensemble model diversity. Further, to adjust the trust region, the KL divergence is replaced by Renyi alpha divergence, whose alpha value is adaptively adjusted during the policy optimization procedure.

1.3 Research Significance

Here, we state the significance of this research work, from the perspective of theory and application, according to the proposed research questions, objectives, and contributions.

1.3.1 Theoretical Significance

This research aims to enhance the performance of reinforcement learning algorithms based on flexible trust region constraints, via adjusting the trust region properly. Our entropy regularization strategy for the KL divergence constraint is to find an alternative and effective way to enhance the exploration ability of trust region policy optimization, with relaxing the trust region from above. Besides bounding the trust region from above, twin trust region policy optimization bounds the trust region from below, which solves the difficulty of searching for an optimal step size for the approximate solution in trust region policy optimization. To measure the trust region for each environment, a linear combination form can effectively adjust and control the balance between two terms in proximal policy optimization, and the parametric alpha divergence provides a new way to measure the trust region. The deep residual attention U-net for dynamics model consists of fewer weights than the widely-used shallow deep neural network, which is typically a lightweight model architecture. Hilbert-Schmidt independence criterion term in the loss function promotes the diversity of ensemble model. Additionally, our adaptive Renyi alpha divergence can adjust the trust region of trust region policy optimization automatically.

1.3.2 Practical Significance

In this research, we apply our proposed reinforcement learning approaches to three benchmark applications, including classic control, MuJoCo, and Box2d. On the one hand, there reinforcement learning algorithms enrich the research contents. On the other hand, their better performance provides an alternative for other benchmark and real-world applications, for example, Atari, autonomous driving, health care, and so on.

1.4 Thesis Organization

This thesis is organized as follows:

Chapter 2: This chapter introduces preliminaries, including Markov decision process, trust region policy optimization, benchmark environments, and performance evaluation.

Chapter 3: In this chapter, we present the literature review on various advances in trust region policy optimization, which is divided into five aspects: taxonomy of existing methods, TRPO and its variants, entropy regularized TRPO, PPO and its variants, and model-based reinforcement learning.

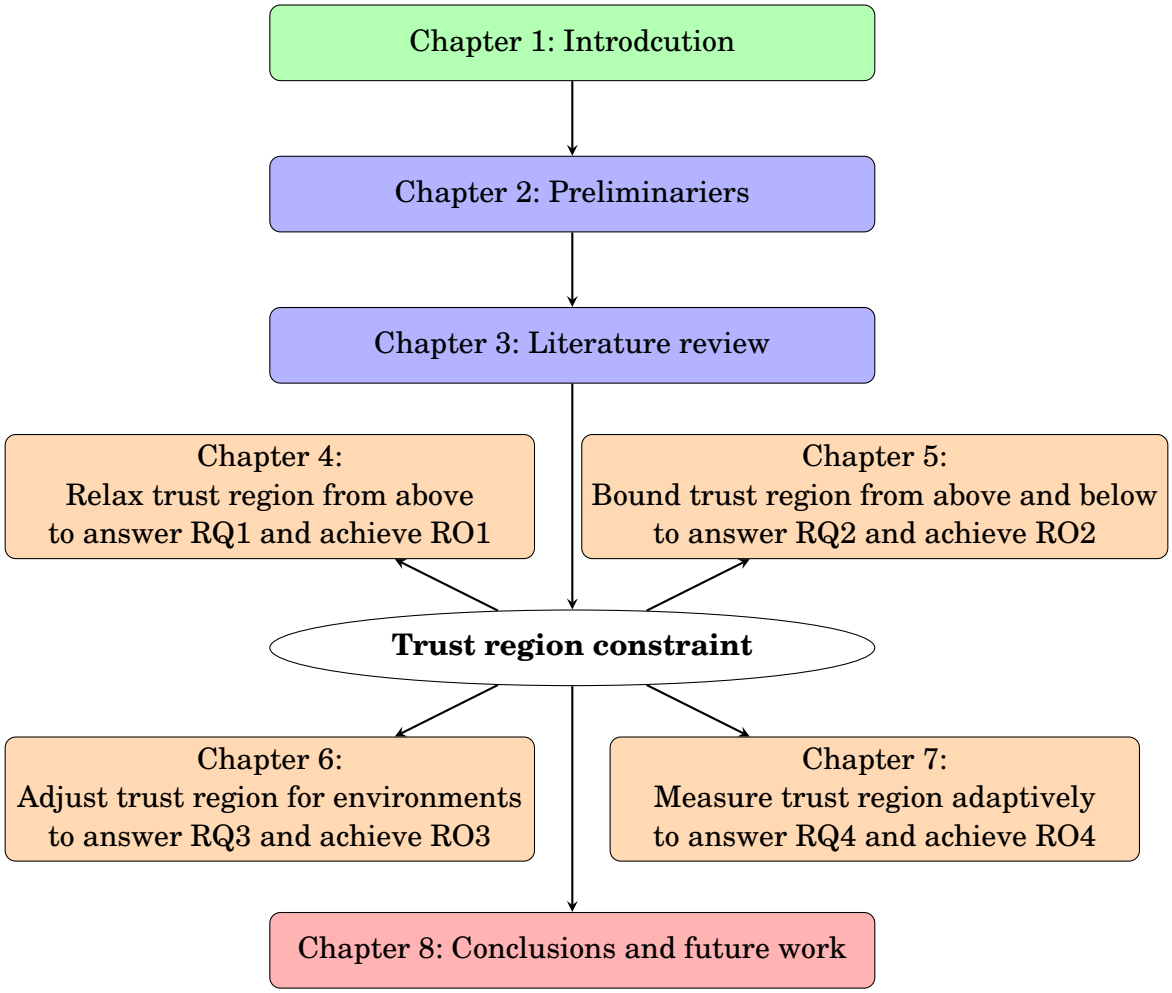


Figure 1.7: The structure of this thesis

Chapter 4: This chapter proposes our trust region policy optimization via regularizing KL divergence constraint. A detailed derivation procedure and a detailed solution are provided, and then our proposed method is validated experimentally. This research answers the research question 1 and achieves the research objective 1. The corresponding research achievement was published as the Paper 2 in the List of Publications.

Chapter 5: In this chapter, we analyze the primary trust region policy optimization and then propose a reciprocal trust region policy optimization. Combining such two algorithms is to build our twin trust region policy optimization. The effectiveness of our proposed technique is illustrated via some detailed experiments. This work answers the research question 2 and achieves the research objective 2, which corresponds to the published Paper 3 and the submitted Paper 5 in the List of Publications.

Chapter 6: This chapter constructs an enhanced proximal policy optimization, which stems from the primary proximal policy optimization and is modified by alpha divergence. Its effectiveness is investigated experimentally. This study answers the research question 3 and achieves the research objective 3. This research outcome is shown in the List of Publications as the published Paper 1.

Chapter 7: In this chapter, we build a concise dynamics model using a true deep neural network architecture, which is based on U-net, residual net and attention mechanism, and Hilbert-Schmidt independence criterion diversity index. On the other hand, an adaptive TRPO is designed and implemented. Our experiments show the superiority of our model-based reinforcement learning. This research answers the research question 4 and achieves the research objective 4, which is also summarized in the submitted Paper 5 of the List of Publications.

Chapter 8: This chapter is to conclude our research work in this thesis and to analyze some future work.

Finally, we summarize the structure of this thesis in Figure 1.7.

CHAPTER 2

PRELIMINARIES

In this chapter, we will provide some preliminaries for reinforcement learning, including Markov decision process, trust region policy optimization, benchmark environments, and performance evaluation.

2.1 Markov Decision Process

Reinforcement learning consists of a discrete time discounted Markov decision process [12, 13, 94, 140] defined by a tuple $\{\mathscr{S}, \mathscr{A}, P, r, \gamma, \rho_0\}$, in which $\mathscr{S}$ and $\mathscr{A}$ respectively denote the state and action spaces of the agent; $P(s'|s,a) : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \mapsto \mathbb{R}$ (one-dimensional real space) is the transition probability distribution from the state $s \in \mathscr{S}$ to the next state $s' \in \mathscr{S}$ after executing an action $a \in \mathscr{A}$; $r(s,a) : \mathscr{S} \times \mathscr{A} \mapsto \mathbb{R}$ is the reward function of the state s and the action a; $\gamma \in (0,1)$ represents the discount factor; and $\rho_0 = P(s_0)$ denotes the initial state distribution [94, 123].

Let $\pi : \mathscr{S} \times \mathscr{A} \mapsto [0,1]$ be a stochastic policy. Given the current state s_t, the agent executes an action $a_t \in \mathscr{A}$ following a conditional probability distribution $\pi(a_t|s_t)$. The agent then receives a reward $r_t = r(s_t, a_t)$. At each time step t, we define a state-action value function $Q^\pi(s_t, a_t)$:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots \sim \pi} \left[\sum_{l=0}^{\infty} \gamma^t r_{t+l} \right] \tag{2.1}$$

and a state value function $V^{\pi}(s_t)$:

$$V^{\pi}(s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots \sim \pi}\left[\sum_{l=0}^{\infty} \gamma^t r_{t+l}\right] = \mathbb{E}_{a_t}[Q^{\pi}(s_t, a_t)]. \tag{2.2}$$

Further, we construct an advantage function:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t). \tag{2.3}$$

These basic mathematical notions will be used to describe our research work.

2.2 Trust Region Policy Optimization

This section outlines TRPO [106] and a detailed solution procedure mathematically. Our research work in this thesis stems from this famous TRPO.

2.2.1 Basic Mathematical Formulation of TRPO

Using a stochastic policy π, the agent collects a trajectory:

$$\tau = \{s_t, a_t\}_{t=1}^{\infty}. \tag{2.4}$$

In this case, the expected discounted reward $J(\pi)$, or the return, becomes

$$\begin{aligned} J(\pi) &= \mathbb{E}_{s_0, a_0, \ldots \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \\ &= \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \\ &= \mathbb{E}_{s_0 \sim \rho_0}[V^{\pi}(s_0)]. \end{aligned} \tag{2.5}$$

Yet, given another policy $\hat{\pi}$, the expected return might be estimated according to the advantage function over π [34, 55, 106], and is then expressed as the following identity:

$$J(\hat{\pi}) = J(\pi) + \mathbb{E}_{\tau \sim \hat{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t)\right]. \tag{2.6}$$

This indicates that if the second term is positive, any policy update could guarantee an improvement to the policy's performance [78, 106]. However, it is difficult to optimize this equation (2.6) since it depends on $\hat{\pi}$. To this end, the relation (2.6) is locally approximated as

$$J(\hat{\pi}) = J(\pi) + \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t)\right] \tag{2.7}$$

which is based on π, rather than $\hat{\pi}$. This ignores any changes in the state visitations resulting from the policy change. Further, via importance sampling [36, 128], the second term in (2.7) can be reformulated as

$$\bar{R}(\hat{\pi}) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\frac{\hat{\pi}(a_t|s_t)}{\pi(a_t|s_t)}A^{\pi}(s_t,a_t)\right]. \tag{2.8}$$

Now, the advantage function $A^{\pi}(s_t,a_t)$ is widely replaced with a generalized advantage estimation (GAE):

$$\hat{A}(s_t,a_t) = \sum_{i=0}^{\infty}(\gamma\lambda)^i\delta_{t+i} \tag{2.9}$$

where δ_t is the temporal difference error:

$$\delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \tag{2.10}$$

and $\lambda\in[0,1]$ is a new added parameter. Here, $\lambda = 0$ indicates a one-step advantage, i.e., $\hat{A}(s_t,a_t) = \delta_t$, while $\lambda = 1$ sums up δ_t over all one-step advantages, i.e., $\hat{A}(s_t,a_t) = \sum_{i=0}^{\infty}\gamma^i\delta_{t+i}$.

Assume that a policy π is parameterized by a set of parameters θ, denoted by π_θ. With the previous policy $\pi_{\theta_{old}}$, the agent interacts with the environment to collect some trajectory data, and then the TRPO defines a concise surrogate objective from (2.8):

$$\hat{R}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}(a_t,s_t)\right]. \tag{2.11}$$

Under these settings, the TRPO gives rise to a constrained maximization problem as follows:

$$\begin{aligned}&\max \hat{R}(\theta)\\ &\text{s.t.}\quad \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(s_t|a_t))]\le\delta\end{aligned} \tag{2.12}$$

where θ and θ_{old} are the parameter vectors corresponding to the current and previous policies, respectively. $D_{KL}(\cdot||\cdot)$ indicates the KL divergence (or relative entropy) [130]; and δ is the threshold of KL divergence. Here, the constraint in (2.12) aims to bound the trust region from above, as shown in Figure 1.2 in Chapter 1.

2.2.2 Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence was introduced in 1951 [62], which is also called relative entropy in information theory. It is a type of statistical measure used to quantify the dissimilarity between two probability distributions and is analyzed in several different views [7, 97, 101, 156].

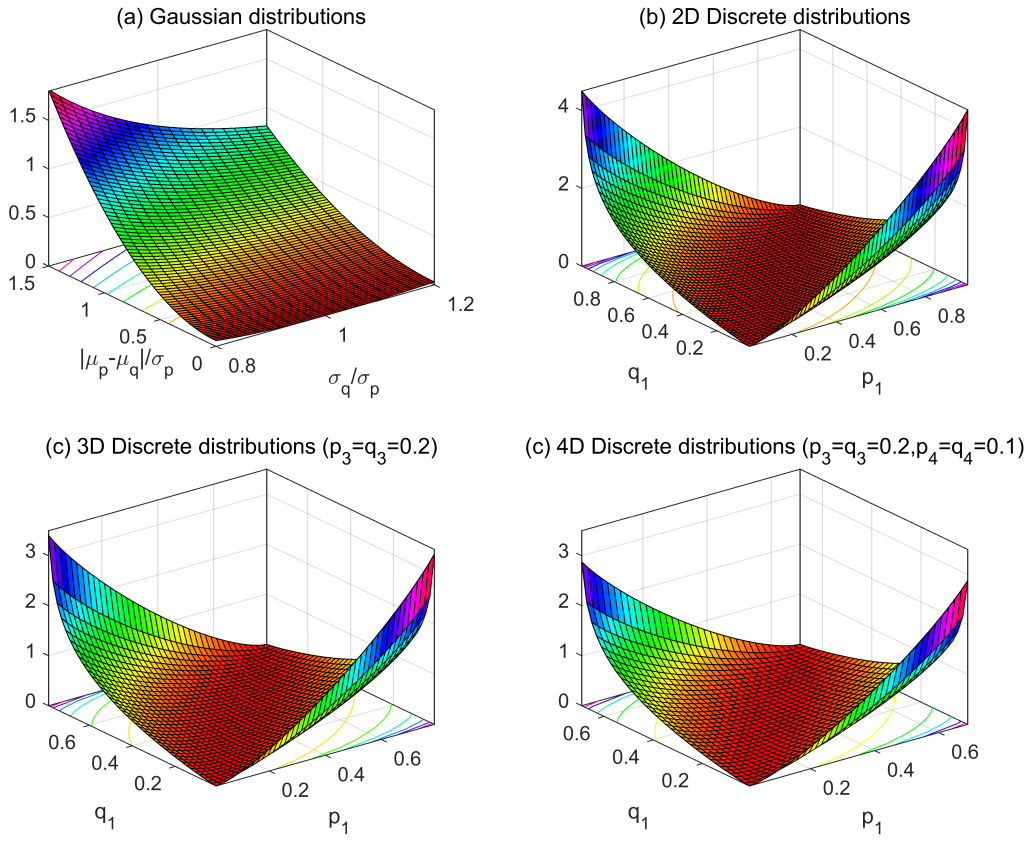


Figure 2.1: KL divergence for Gaussian and discrete distributions

The KL divergence for two continuous random distributions ($p(x)$ and $q(x)$) is defined as

$$D_{KL}(p||q) = \int p(x) \ln \frac{p(x)}{q(x)} dx. \tag{2.13}$$

In reinforcement learning, it is usually assumed that continuous actions satisfy a Gaussian distribution. Given two univariate Gaussian distributions, i.e., $p(x) \sim N(\mu_p, \sigma_p^2)$ and $q(x) \sim N(\mu_q, \sigma_q^2)$, this KL divergence will have a closed form:

$$D_{KL}(p||q) = \frac{1}{2}\left(\ln \frac{\sigma_q^2}{\sigma_p^2} + \frac{\sigma_p^2 + (\mu_p - \mu_q)^2}{\sigma_q^2} - 1 \right). \tag{2.14}$$

Further, we consider that the n continuous actions ($x = [x_1, ..., x_i, ..., x_n]^T$) satisfy a n-variate Gaussian distribution with a mean vector $\mu = [\mu_1, \mu_2, ..., \mu_n]^T$ and a diagonal co-

variance matrix (i.e., $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_n^2)$. In this case, the KL divergence is simplified to

$$\begin{aligned} D_{KL}(p||q) &= \sum_{i=1}^{m} \frac{1}{2}\left(\ln \frac{\sigma_{q_i}^2}{\sigma_{p_i}^2} + \frac{\left(\mu_{p_i} - \mu_{q_i}\right)^2}{\sigma_{q_i}^2} + \frac{\sigma_{p_i}^2}{\sigma_{q_i}^2} - 1\right) \\ &= \sum_{i=1}^{m} D_{KL}(p_i||q_i) \end{aligned} \tag{2.15}$$

where $p(x_i) \sim N(\mu_{pi}, \sigma_{pi}^2)$ and $q(x_i) \sim N(\mu_{qi}, \sigma_{qi}^2)$. This also shows that KL divergence is an additive model.

For two discrete distributions with n different probabilities ($p = [p_1, ..., p_i, ..., p_n]^T$ and $q = [q_1, ..., q_i, ..., q_n]^T$), the KL divergence is defined as

$$D_{KL}(p||q) = \sum_{i=1}^{n} p_i ln \frac{p_i}{q_i}. \tag{2.16}$$

In Figure 2.1, we illustrate the KL divergence for Gaussian and three discrete distributions. Further, it has been proven in [27, 62, 76, 130] that the KL divergence has the following useful properties:

(a) **Convexity**: $D_{KL}(p||q)$ is convex with respect to both p and q;

(b) **Strict positivity**: $D_{KL}(p||q) \geq 0$ and $D_{KL}(p||q) = 0$ if and only if $p = q$.

2.2.3 The Solution to TRPO

Engstrom et al. [37] point out that code-level optimization has a major impact on agent behavior in TRPO. Thus, in this sub-section, we provide a detailed procedure to solve the optimization problem (2.12) of TRPO that follows the complementary material for TRPO [106] and related reference [34]. Traditionally, the optimization problem (2.12) is converted into a constrained minimization problem:

$$\begin{aligned} &\min \ -\hat{R}(\theta) \\ &\text{s.t.} \quad \hat{D}_{KL}(\theta) \leq \delta \end{aligned} \tag{2.17}$$

where

$$\begin{aligned} \hat{R}(\theta) &= \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}(s_t, a_t)\right] \\ \hat{D}_{KL}(\theta) &= \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))]. \end{aligned} \tag{2.18}$$

At first, $\hat{R}(\theta)$ is approximated linearly around θ_{old}, i.e.,

$$\begin{aligned} \hat{R}(\theta) &\approx \hat{R}(\theta_{old}) + (\theta - \theta_{old})^T \nabla \hat{R}(\theta_{old}) \\ &= \hat{R}(\theta_{old}) + g^T d \end{aligned} \tag{2.19}$$

where the descent direction d becomes

$$d = \theta - \theta_{old} \tag{2.20}$$

and the gradient of $\hat{R}(\theta)$ with respect to (wrt) θ is

$$g = \nabla \hat{R}(\theta_{old}). \tag{2.21}$$

The constraint is approximated quadratically, i.e.,

$$\begin{aligned} \hat{D}_{KL}(\theta) \approx & \hat{D}_{KL}(\theta_{old}) \\ & + (\theta - \theta_{old})^T \nabla \hat{D}_{KL}(\theta_{old}) \\ & + \tfrac{1}{2}(\theta - \theta_{old})^T \nabla^2 \hat{D}_{KL}(\theta_{old})(\theta - \theta_{old}) \\ = & \tfrac{1}{2} d^T H d \end{aligned} \tag{2.22}$$

where the Hessian matrix $H = \nabla^2 D_{KL}(\theta_{old})$ is called a Fisher information matrix. In term of the properties of KL divergence [27, 62, 76, 130], we have $D_{KL}(\theta_{old}) = 0$, and $\nabla D_{KL}(\theta_{old}) = 0$.

Now, we obtain a linear-quadratic programming:

$$\begin{aligned} & \min \; -g^T d \\ & \text{s.t.} \quad \tfrac{1}{2} d^T H d \le \delta. \end{aligned} \tag{2.23}$$

In order to estimate the descent direction d, the above constrained problem is reformulated as an unconstrained problem via a Lagrangian multiplier technique:

$$L(d) = -g^T d + \lambda \left(\frac{1}{2} d^T H d - \delta \right). \tag{2.24}$$

Let the gradient of $L(d)$ be zero with respect to d, and then we have a set of linear equations:

$$Hd = \frac{1}{\lambda} g \tag{2.25}$$

and a search direction u formally:

$$u = H^{-1} g = \lambda d = \frac{d}{\eta} \tag{2.26}$$

after defining the step size $\eta = 1/\lambda$.

According to the constraint in (2.23), i.e.,

$$\frac{1}{2}(\eta u)^T H (\eta u) = \frac{1}{2} \eta^2 u^T H u \le \delta \tag{2.27}$$

we have a maximum η_{max} for the step size η:

$$\eta_{\max} = \sqrt{\frac{2\delta}{u^T H u}}. \tag{2.28}$$

Further, since $\hat{R}(\theta)$ can not be proven to be concave, a linear search is executed to find out the optimal step shrinking index i^*, so as to ensure an improvement to the return $\hat{R}(\theta)$:

$$\begin{aligned} &\max \hat{R}(\theta_{old} + \eta_0^i \eta_{max} u) \\ &\text{s.t. } \hat{D}_{KL}(\theta_{old} + \eta_0^i \eta_{max} u) \leq \delta \\ &\quad i = 0, ..., n \\ &\quad \eta_0 \in (0, 1) \end{aligned} \tag{2.29}$$

where n is the number of linear searches, e.g., $n = 15$, and η_0 is the shrinking factor. Finally, the θ vector is updated as follows:

$$\theta = \theta_{old} + \eta_0^{i^*} \sqrt{\frac{2\delta}{u^T H u}} u \tag{2.30}$$

which is used to train the actor in TRPO.

The critic is to minimize the following temporal difference error:

$$\min \hat{\mathbb{E}}_t \left[\frac{1}{2} (r_t + \gamma V(s_{t+1}) - V(s_t))^2 \right]. \tag{2.31}$$

Since this is an unconstrained minimization problem, it can be optimized by many widely-used gradient-based optimizers, for example, adaptive moment estimation (ADAM) [56].

2.2.4 Conjugate Gradient Algorithm for Search Direction

In order to avoid calculating the inverse matrix H in (2.26), the linear set of equations $Hu = g$ can be solved via the conjugate gradient method [42]. This method converts $Hu = g$ into a minimization problem:

$$\min \frac{1}{2} ||Hu - g||_2^2 = \frac{1}{2} u^T H u - g^T u \tag{2.32}$$

which then is optimized via gradient descent method and conjugate gradient idea [42].

If $H \in \mathbb{R}^{n \times n}$ is symmetric, real, and positive definite, $g \in \mathbb{R}^n$, and $Hu_0 \approx g$, then the conjugate gradient method [42] computes an optimal solution u^* such that $Hu^* = g$, as summarized in **Algorithm 1**. Given the matrix H, the time complexity of each

Algorithm 1 Conjugate gradient method

1: INPUT: H, g and u_0
2: PROCESS
3: $k = 0$
4: $r_0 = g - Hu_0$
5: while $r_k \neq 0$
6: $k = k + 1$
7: if $k = 1$
8: $p_1 = r_0$
9: else
10: $\beta_k = \frac{r_{k-1}^T r_{k-1}}{r_{k-2}^T r_{k-2}}$
11: $p_k = r_{k-1} + \beta_k p_{k-1}$
12: end
13: $\alpha_k = r_{k-1}^T r_{k-1} - \frac{1}{p_k^T H p_k}$
14: $u_k = u_{k-1} + \alpha_k p_k$
15: $r_k = r_{k-1} - \alpha_k H p_k$
16: end
17: OUTPUT $u^* = u_k$

iteration (i.e., from the while (Line 5) to its end (Line 16)) is $O(n^2)$, since the most costly computation stems from calculating the $p_k^t H p_k$.

Notably, however, it will be costly to compute and store the matrix H with some large scale applications. In practice, only the vector Hu (i.e., Hp_k) is used frequently in **Algorithm 1**, which is equal to the gradient of the KL divergence and is estimated directly. Computing this full gradient needs a time complexity of $O(n)$ [22]. In this case, the time complexity of each iteration is reduced into $O(n)$. Moreover, it is only to execute several iterations to achieve a satisfactory performance, for example, $m = 10$ iterations. Therefore the overall time complexity is $O(mn)$, which implies that the scalability of the conjugate gradient method is very good. Additionally, the space complexity is only $O(n)$ without storing the matrix H.

2.2.5 Trust Region Policy Optimization Algorithm

The TRPO consists of two networks: actor and critic, which are structured as deep neural networks. For the state vectors, both of them are implemented by multi-layer perceptron networks with ReLU activation function [2, 43], as shown in Figures 2.2 and 2.3. Generally speaking, the same architectures are designed except for the output layers.

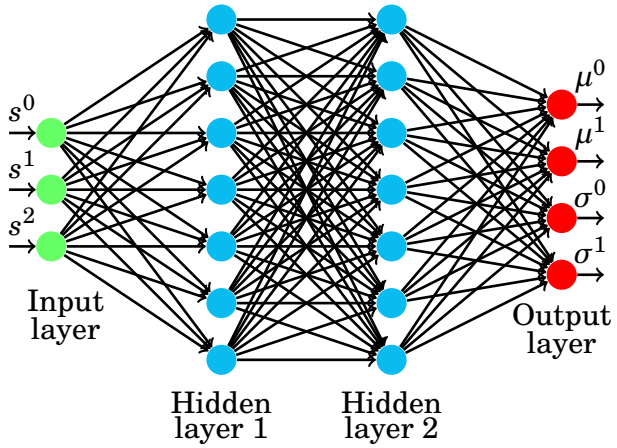


Figure 2.2: The actor architecture, where $|\mathscr{S}| = 3$ and $|\mathscr{A}| = 2$.

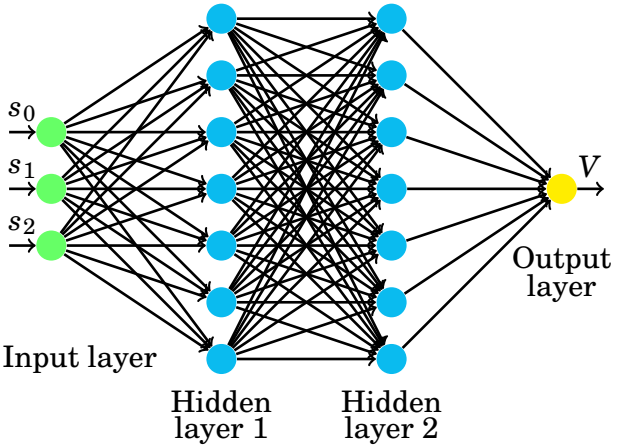


Figure 2.3: The critic architecture for the state value function, where $|\mathscr{S}| = 3$ and $|\mathscr{A}| = 2$.

On the other hand, these two networks correspond to the policy $\pi(a_t|s_t)$ and the state function $V(s_t)$, respectively. This representation means that the actor and critic are parameterized by the network weights (and bias terms), usually indicated by θ and ϕ, respectively.

At last, we summarize the pseudo code for trust region policy optimization in **Algorithm 2**.

2.3 Summary

In this chapter, we introduced Markov decision process and trust region policy optimization, which will be used in the next several chapters to support our research in this

Algorithm 2 The pseudo code of TRPO

INPUT
    A specific experimental environment
    An actor network architecture
    A critic network architecture and its learning rate
    A reward discount factor γ
    A step size shrinking factor η_0
    A generalized advantage estimation factor λ
    The number of episodes N
    A random seed
INITIALIZATION
    Initialize the actor and critic networks
    Set a random seed in the environment
PROCESS
    Execute the following procedure till reaching N
        Collect trajectory data by running the current policy π_θ in the environment
        Calculate the generalized advantage estimation $\hat{A}(s_t, a_t)$
                across the entire trajectory
        Update the actor policy network as follows
            Calculate the gradients for the surrogate objective: g
            Solve a set of linear equations to generate: u
            Execute a linear search for the step size: η
            Update the weights of the actor network θ
        Update the critic value network by optimizing (2.31)
OUTPUT
    The trained actor and critic networks

thesis.

CHAPTER 3

LITERATURE REVIEW

In the above chapter, we provide some preliminaries for trust region policy optimization. In this chapter, we will review some related work for trust region policy optimization reported in the literature after a taxonomy of reinforcement learning algorithms is summarized.

3.1 Taxonomy of Reinforcement Learning Algorithms

Nowadays, there are many reinforcement learning algorithms, which can be categorized according to different viewpoints. Here we only introduce several categorizations associated with our research.

Offline vs online. The offline RL is to learn exclusively from static datasets of previously collected interactions, while the online RL is to interact with the actual environments to collect some trajectories in real time way [93]. In this thesis, we focus our study on online setting.

On-policy vs off-policy. Under on-policy situation, an agent is free to interact with the environment and must collect a new set of experiences after every update to its policy. For off-policy cases, besides some real time interactions with the environment, a replay buffer is used to store some experienced data, from which some collected data are extracted to update its policy [34, 123].

Value optimization and policy optimization. In on-policy RLs, an agent aims to improve its policy to achieve a higher reward in the process named policy optimization,

which is based on two strategies: value-based optimization and policy-based optimization, and further are implemented by critic and actor architectures, respectively. The former optimizes an action-state value function to produce a probability of selecting an action. The latter optimizes the policy directly according to the sampled rewards. Currently, such two strategies are combined to form a hybrid one, implemented by actor-critic architecture [34, 123].

Model-free vs Model-based. Here, the model implies an environment dynamics model to represent the transition probability distributions (and rewards). In model-free RL, it is assumed that each environment has its existing dynamic model to emit the transition probability and reward. In model-based RL, such a model needs to be trained via a small number of interactions with the environment, which produces some pseudo trajectories to update the policy. It is widely recognized that model-based RL techniques can achieve a higher sample efficiency than model-free ones. [69, 92].

According to the above categorization, our research in this thesis belongs to two aspects: on-policy actor-critic RL methods and model-based on-policy RL techniques. In the following sections, we will review some related work associated with our research.

3.2 Trust Region Policy Optimization and its General Variants

The original TRPO is a typical on-policy reinforcement learning technique, which has been extended to create several on- and off-policy variants. In [123], it is pointed out that, on-policy methods attempt to evaluate and improve the policy that is used to make decisions, whereas off-policy methods evaluate and improve a policy being different from that used to generate the data. In practice, what differentiates between both of them is that a replay buffer is used to store some experienced data for off-policy techniques particularly.

3.2.1 On-policy TRPO Variants

Kronecker-factored approximate curvature (K-FAC) is an efficient method for approximating the natural gradient descent in neural networks [73]. In Wu et al. [143], the researchers modified the K-FAC to optimize actor-critic with a trust region to result in an actor-critic method with Kronecker-factored trust region (ACKTR).

Most continuous action methods assume a Gaussian distribution with a diagonal covariance matrix. However, some researchers have looked to improve the performance of the reinforcement learning techniques by investigating more complicated probability estimations. For example, normalizing flows are a class of generative models that transform a simple probability distribution, such as a normal Gaussian distribution, into a more complicated distribution via a sequence of invertible and differentiable mappings [59]. Tang and Agrawal [125] boosted TRPO by normalizing this flow policy, while Terpin et al [126] explored optimal transport discrepancies to define which trust regions to enhance in a method called OT-TRPO.

To investigate the global convergence of TRPO as well as PPO (proximal policy optimization) [107], Liu et al. [67] constructed a variant of TRPO and PPO with an over-parameterized neural network. The model converges globally at a sub-linear rate to provide optimal policy updates.

Compared to TRPO, Quasi-Newton trust region policy optimization (QNTRPO) [54] modifies three aspects of TRPO: i) A quadratic approximation via a Quasi-Newton approximation of the Hessian matrix is used to replace the inverse matrix of Hessian matrix; ii) The Dogleg method is used to define the step size; and iii) The adaptive change of the step size is adjusted by the classical trust region constraint.

Because TRPO's solution is approximate it is sometimes not possible to find an optimal solution within the trust region. Hence, Otto et al. [89] proposed differentiable neural network layers to enforce a trust region specially for deep Gaussian policies via closed form projections.

To enhance the robustness and adaptiveness of the agent, Queenty et al. [96] developed an uncertainty-aware trust region policy optimization (UA-TRPO), which controls finite-sample estimation errors in two key components of TRPO: (a) the policy gradient which directly considers the uncertainty in the policy gradient estimate; and (b) the trust region metric which restricts policy updates to a subspace of the trust region via the observed trajectories.

Additionally, more complicated solutions have been devised. For example, Zhao et al. [160] solved the trust region subproblem via a preconditioned stochastic gradient method and a linear search scheme that ensures each step contributes to the model's function and stays in the trust region. Based on mirror descent [11], Shani et al. [109] considered a sample-based TRPO, which converges quickly to the global optimum.

Some more complicated situations have been also investigated. For example, Azizzadenesheli et al. [8] presented a solution for generalized trust region policy optimization

(GTRPO). This method copes with both the Markov decision processes and any partially observable Markov decision process to guarantee that the policy updates monotonically improve the expected cumulative rewards. Li and He [65] put forward the concept of multiagent TRPO. And, Zhang and Roos [157] considered averaged reward rather than the more widely-used discounted reward within an infinite-horizon environment. For safe reinforcement learning, Zhang et al. [155] considered a safety criterion as a constraint on the conditional value-at-risk (CVaR) of accumulative costs, and then proposed a CVaR-constrained policy optimization to maximize the expected return under a limited upper tail of constrained costs. As a byproduct, a trust region method for CVaR constrained MDPs is introduced, which essentially adds a new constraint: CVaR into TRPO. However, no experiment is reported in [155].

What all these variants of TRPO have in common is that they are constrained forms of TRPO. However, the proximal policy optimization (PPO) algorithm [107] converts TRPO into an unconstrained optimization problem via a technique involving Lagrangian multiplier. This new optimization problem is solved through a first-order gradient optimization. Further, the clipping variant of PPO uses a clipping function to replace the KL divergence, which indirectly restricts the trust region [34].

3.2.2 Off-policy TRPO Variants

Since the original on-policy TRPO requires a large amount of on-policy interaction with the environment. Several off-policy variants of TRPO have been investigated. These variants generally improve sample efficiency.

In the sparse reward variant, the agent is only rewarded when it reaches the desired goal. This negates the need to design a nuanced reward mechanism. The concept of hindsight is also an attractive idea, which refers to the algorithm's ability to learn from information across goals, including past goals not intended for the current task. Purposing hindsight to sparse rewards, Zhang et al. [154] proposed a hindsight TRPO, i.e., HTRPO, which introduces a hindsight goal filtering strategy to cope with sparse reward applications. Lastly, Meng et al. [78], outlined an off-policy TRPO, where the surrogate objective is based on both on-policy and off-policy data. The solution guarantees a monotonic improvement further theoretically.

3.3 Entropy Regularized Trust Region Policy Optimization

Entropy regularized TRPOs and their variants add a proper entropy regularization term [29, 83, 127] to their objectives. This is believed to help with exploration because it encourages the agent to select policies more randomly [3], and hence the agent's performance improves. As in the last section, we also divide these variants into two sub-categories: on-policy and off-policy.

3.3.1 On-policy Entropy Regularized TRPOs

Roosraie and Ebadzadehis variant [103] adds a regularizer to the surrogate objective, while in Eysenback and Levineis work [38], the regularizer is added to the expected cumulative rewards without a discount factor. Ma [71], however, adds entropy and log probability regularization terms to the discounted rewards. In Cayci et al. [20], the value function is regularized and approximated linearly using a softmax parameterization, in which no constraint is considered. Similarly, the entropy regularization can be applied to the value functions to yield fast convergence [21]. Akrour et al. [4] treat entropy regularization as a second constraint. They then use a set of projections to transform their constrained problem into an unconstrained one. However, notably, the subsequent optimization procedure is rather complicated. Ahmed et al [3] experimentally observed that adding entropy regularization results in a smooth objective which, in turn, leads to faster convergence.

3.3.2 Off-policy Entropy Regularized TRPOs

Path consistency learning (PCL) [84] minimizes the notion of soft consistency errors along multi-step action sequences extracted from both on- and off-policy trajectories. In [85], the research team combined PCL with a trust region strategy. Hence, an off-policy trust region method (i.e., TRUST-PCL) for continuous control was proposed, which adds a discounted entropy regularization term to the surrogate objective. Soft actor critic (SAC) [48] is a representative off-policy reinforcement learning algorithm based on maximum entropy that adds an entropy regularizer to the expected reward, thus building a linearly composite objective.

3.4 Proximal Policy Optimization and its Variants

In the aforementioned two sections, TRPO and its various variants need a second-order optimization approach, which evaluates a Hessian matrix at each update step. Proximal policy optimization (PPO) [107] converts the constrained optimization in TRPO into an unconstrained one, which is solved using a first-order optimization technique, that is, a gradient vector is calculated at each update step. Now, there are three main forms of PPO: primary, adaptive, and clipping [107]. In this section, we will review PPO and its main improvements so far.

3.4.1 PPOs Variants with Trust Region

The primary PPO is formulated as an unconstrained optimization, i.e., accumulative discount return term plus KL divergence, via a penalty factor to control the trade-off between these two terms. In PPO-lambda [23], a single penalty factor is adaptively adjusted. To the best of our knowledge, this is the sole modification of primary PPO.

The adaptive PPO adjusts the penalty factor dynamically during its policy update, with three additional tunable parameters, besides the original tunable factor. In [35], two early stopping optimization techniques for an adaptive PPO variant are investigated, KLE-Stop stops the policy update within an epoch when the mean KL divergence exceeds a pre-defined threshold, and KLE-Rollback rolls back the policy before the update iterations. Recently, this adaptive PPO has been extended to a continuous time/space situation, which corresponds to a continuous PPO (CPPO) with adaptive penalty constant [159], for continuous reinforcement learning.

3.4.2 PPOs Variants without Trust Region

The clipping uses a clipping factor to replace both the KL divergence and the tuning factor, which has become the most popular form in the literature. For this clipping PPO, it is argued whether the clipping function could act as the penalty term as in the primary and adaptive PPOs. In [135], two improvements are presented to construct a modified PPO (TR-PPO-RB), where a new clipping function is proposed to support a rollback behavior to restrict the probability ratio of the current policy to the previous one (PPO-RB), and the KL divergence is calculated to enforce the trust region constraint (TR-PPO) outside of each PPO update. Further, this PPO-RB with a sloped clipping function is improved by a curved tanh function in [163]. In trust region-guided PPO (TRGPPO) [136],

the clipping range is adjusted adaptively by the trust region to improve the exploration behavior of the clipping PPO. In [119], an early stopping policy optimization (ESPO) is proposed, which uses the surrogate objective in the clipping PPO without any ratio clipping and considers an indicator function to estimate an optimal time point to drop out from the multiple optimization epochs rather than a fixed number of optimization epochs in the clipping PPO. A fast proximal policy optimization (FastPPO) is proposed in [161], which replaces the primary minimum operation with two accelerating operations: linear pulling and quadratic pulling. In simple policy optimization (SPO) [145], to characterize the trust region better, a novel clipping function is integrated into PPO to describe the KL divergence between two adjacent policies.

To improve the exploration ability and to deal with the possible negative advantage estimation, an upper confidence bound advantage function PPO is presented (UCB-AF) [144], which calculates the confidence of the advantage estimation according to Hoeffding's inequality, and adjusts the advantage estimation with an upper confidence bound. By introducing an alpha-policy to stand as a locally superior policy, gradient informed proximal policy optimization (GI-PPO) [114] incorporates analytical gradients from differential environments into PPO framework, in which by adaptively modifying the alpha value the influence of analytical policy gradients can be managed in the learning process.

As a representative actor-critic architecture, the critic network provides the value function estimation for the actor network, but the latter network does not link to the former inversely. In PPO with policy feedback (PPO-PF) with a similar objective in clipping PPO [46], the policy from the actor guides the critic update, resulting in a collaborative update procedure for the two networks.

To deal with more complicated RL settings, in [159] on safe reinforcement learning, a safety criterion defined as the conditional value-at-risk (CVaR) of cumulative costs is considered, and a CVaR-constrained policy optimization is proposed, which can be regarded as a safe version of clipping PPO. For the infinite horizon environments, under the average reward setting, PPO is extended to average reward PPO in [72] for the long-run performance evaluation.

The above improvements on clipping PPO are still under on-policy situation. To enhance sample efficiency and exploration ability, an off-policy advantage function is calculated via the reuse of previous policies, and then a proximal policy optimization with advantage reuse (PPO-AR) is first constructed [24]. Subsequently, two variants are considered, where proximal policy optimization with advantage reuse of competition

(PPO-ARC) is designed to promote the sample efficiency, and proximal policy optimization with generalized clipping (PPO-GC) is aimed to change the policy flat clipping boundary.

3.5 Different Divergences for Trust Regions

To measure the difference between two policies, i.e., the trust region, besides KL divergence, various divergences or distances are investigated in the reinforcement learning community.

In TRPO [106], the original divergence is a total variance divergence (TV), which is essentially equal to half the absolute distance between two discrete distributions. Then, this TV is replaced by KL divergence [130], as an upper bound of TV. In [14], f-divergence constrained policy improvement is introduced, the algorithms of which are very similar to TRPO. It is worth noting that alpha divergence belongs to the f-divergence class as stated in [7]. In divergence actor-critic (DivAC), the Renyi alpha divergence is verified, which limits non-negative alpha values [150]. The above divergences all serve to measure two action distributions. However, in divergence augmented policy optimization (DAPO) [132], the Bregman divergence [7] is estimated for two state distributions of two policies, which is used in a novel class of mirror descent methods [52, 148].

The original PPO [107] considers the KL divergence [130]. In [25], a squared difference between two distributions is defined as a point probability distance, which is very similar to a squared Euclidean distance and becomes a lower bound of TV, to build a policy optimization with point probability distance (POP3D). The raw Pearson (PE) divergence is proposed in [146], which is generalized to construct a relative PE divergence in PPO-RPE in [58]. The correntropy [68] was introduced for non-Gaussian signal processing, which replaces KL divergence to build CIM-PPO in [47]. In order to combine off-policy advantages into on-policy algorithms, generalized PPO with sample reuse (GePPO) [95] comes back to the original TV in TRPO.

3.6 Model-based Reinforcement Learning with Trust Region

Both TRPO and PPO belong to model-free reinforcement learning [69, 81, 92], which executes numerous interactions with the true environment to achieve a satisfactory performance. This results in lower sample efficiency. To cope with this limitation, model-

based reinforcement learning has been considered [69, 81, 92], which aims to learn an environment's dynamics model with a reduced amount of interactions with the true environment. Further, ensemble learning [63, 77, 99] is applied to enhance the learned models. In this section, we mainly review related work on model-ensemble-based reinforcement learning.

During 1990 and 1991, the RL pioneer Sutton proposed the first model-based RL algorithm Dyna [121, 122, 124], in which the training procedure iterates between two key steps. First, using the current policy, some trajectory data is gathered from interaction with the environment, which then is used to learn the dynamics model. Second, the policy is improved with imagined data generated by the learned dynamics model. This work established a famous Dyna-style framework for MBRL. Plaat et al. [92] generalize this framework to simplify MBRLs into two aspects: learning and planning. The former is to learn a local transition dynamics model to enhance the sample efficiency of the policy learning, under a supervised learning paradigm. The latter is to apply a planning algorithm (e.g., policy optimization or model predictive control (MPC)) to improve the behavior policy.

It is usually found that policy optimization tends to exploit regions where insufficient data is available to train the model, leading to catastrophic failures. This issue is referred to as model bias [32], and occurs relatively frequently in some single model learning methods [81, 92]. To reduce this bias, some researchers have shifted their interests to ensemble learning [77, 99], which builds a set of dynamics models and then integrates them into a composite dynamics model, which is regarded as model ensemble RL or simply MERL. In this study, we will pay more attention to these state-of-the-art ensemble models based on deep learning techniques.

The first algorithm is model-based trust region policy optimization (ME-TRPO) [63], which uses an ensemble of deterministic neural networks to model the environment dynamics, without a reward function. This ensemble is trained by minimizing the squared L2 error loss, using some samples collected from the true environment. It is worth noting that these networks only differ by the initial weights and the order in which mini-batches are sampled. In the policy learning step, the policy is updated using TRPO [106] according to fictitious samples generated by one of the learned dynamics models chosen at random.

In principle, stochastic lower bounds optimization (SLBO) [70] is a variant of the above ME-TRPO with theoretical guarantees of monotonic improvement. In practice, instead of using a single-step squared L2 error loss, SLBO uses a multi-step L2-norm

loss to train the dynamics models. Additionally, the dynamics models are trained using some noisy samples, the update for dynamics and policy is repeated several times and the entropy regularization term is added to the objective of TRPO.

Probabilistic ensembles with trajectory sampling (PETS) [26] combines uncertainty-aware deep network dynamics models with sampling-based uncertainty propagation. In the PETS algorithm, the dynamics are modeled by an ensemble of probabilistic neural network models, which captures both epistemic uncertainties from limited data and network capacity, and aleatoric uncertainty from the stochasticity of the ground truth dynamics. Note that these networks are trained with identical size subsets sampled from the collected trajectory data randomly with replacement. Combining random shooting (RS) [98] and cross-entropy method (CEM) [17] from model predictive control (MPC), respectively, this technique induces two detailed algorithms: PETS-RS and PETS-CEM. It is worth noting that both MPC approaches could result in a relatively high computational complexity.

Model-based policy optimization (MBPO) [53] can be regarded as a variant of PETS with theoretical guarantees of monotonic improvement. To generate a prediction from the ensemble, a model is selected uniformly at random, allowing for different transitions along a single model rollout to be sampled from different dynamics models. MBPO adopts off-policy soft actor-critic (SAC) [48] as its policy optimization algorithm. In its model usage, the branching strategy is applied, in which model rollouts begin from the state distribution of a different policy under the true environment dynamics. There are two improved versions of MBPO. Masked model-based actor-critic [90] reduces the influence of model error with a masking mechanism that trusts the model when it is confident and eliminates unreliable model-generated samples. Conservative model-based actor-critic (CMBAC) [137] learns multiple estimates of the Q-function from the ensemble models and averages the bottom-k estimates to optimize the policy.

Note that there are two ensemble strategies: averaging over a set of dynamics models and selecting one of the trained models, which are possibly used in different available software packages in Github.

We summarize these main characteristics of the four MERL algorithms in Table 3.1, in which TRPO is a mainstream algorithm for policy optimization.

Table 3.1: Main characteristics of existing model ensemble RL methods

Method	Learning	Planning
ME-TRPO	**Deterministic function**: 2/4 layer deep neural networks (Squared L2 error loss, different initial weights, identical sample subsets) **Model ensemble**: Select one randomly	TRPO (on-policy)
SLBO	**Deterministic function**: Deep neural networks (L2 error loss, different initial weights, identical sample subsets)	TRPO with entropy regularization (on-policy)
PETS	**Stochastic function**: 3 layer probabilistic neural networks (Negative log probability loss, different subsets sampled with replacement)	RS CEM (MPC)
MBPO	**Stochastic function**: probabilistic neural networks (L2 error loss and variance loss different initial weights, identical sample subsets)	SAC (off-policy)

3.7 Research Gaps Identified in Literature Review

According to the aforementioned review on literature, we summarize the following research gaps:

- In reinforcement learning, it is of importance to control the trade-off between exploration and exploitation. The primary trust region policy optimization is often criticized to compress its exploration ability due to its strict constraint on two adjacent policies. To this end, it was widely to add the Shannon entropy to its (surrogate) objective, to promote the exploration ability. However, this is an indirect strategy to search for some suspicious policies. In this thesis, we argue whether it is possible to find out a direct way to embed this Shannon entropy. This is the first gap from an indirect entropy regularization to a direct one, identified in our literature review.

- From an optimization technique viewpoint, a successful approximate solution to a non-linear constrained problem is associated with two key factors: search direction and step size, in which the latter needs a proper interval. The existing solution to trust region policy optimization only provides an upper bound of step size, but is short of a lower bound, which results in a negative impact on an effective solution to trust region policy optimization. The second gap to be bridged is to find out an appropriate lower bound to construct an interval of step size for trust region policy optimization.

- The most popular version of proximal policy optimization is its clipping form, which uses a clipping function to replace the KL divergence to measure the difference between two successive policies indirectly. It seems necessary to revisit the primary version which covers the KL divergence to characterize two adjacent policy difference directly. Additionally, we argue whether the distinct environments need different divergences to reflect their policy difference effectively. This induces the third gap from the clipping version to the primary one for proximal policy optimization and from the KL divergence to a parameterized divergence.

- It is widely recognized that the ensemble model-based reinforcement learning can enhance the sample efficiency to some extent, which consists of two iterative steps: building an ensemble dynamics model and learning a policy. The existing dynamics models are simply assembled using several deep neural networks, without considering an explicit model diversity. When the policy is trained via trust region policy optimization, it is also argued whether a parameterized divergence form can work better. This results in the fourth gap from the simple ensemble dynamics model to a light-weight and diverse ensemble dynamics model, and from the KL divergence to a more complicated parameterized divergence.

In the next four chapters, we will bridge or narrow these four gaps, which will improve the performance of reinforcement learning algorithms and moreover enrich the achievements in related reinforcement learning areas.

3.8 Summary

In this chapter, we reviewed some related work on trust region policy optimization, its variant proximal policy optimization, and its model-ensemble-based techniques, which will provide a good foundation for our research work in this thesis.

CHAPTER 4

ENTROPY REGULARIZED CONSTRAINT TRUST REGION POLICY OPTIMIZATION

In this chapter, we will extend trust region policy optimization via applying an entropy regularization to the Kullback-Leibler divergence constraint.

4.1 Introduction

In Chapter 2, we review the TRPO [106] detailedly, which maximizes a surrogate objective associated with a generalized advantage function and importance sampling, when the KL divergence between two adjacent policies is lower than a pre-determined threshold. The scalability and efficiency of TRPO approach have seen it successfully taken up in many applications. However, it has a strict constraint, which restricts the updating policy and its previous one in the pre-defined trust region. For some application environments, such a strict constraint possibly excludes all suspicious gradients. This typically stops the agent from fully exploring the environment and hurting the performance [64, 71].

One way to encourage exploration is to insert Shannon entropy [29] or one of its related variants (e.g., log probability [71]) into the expected reward or surrogate objectives. For example, in TRPO variant [103], entropy regularization is added to the surrogate objective. Similarly, Ma [71] integrated entropy into reward, and log probability into state value function and state-action value functions, to improve TRPO. In this study, we refer to this kind of entropy-regularized objective TRPO as ERO-TRPO. To the best

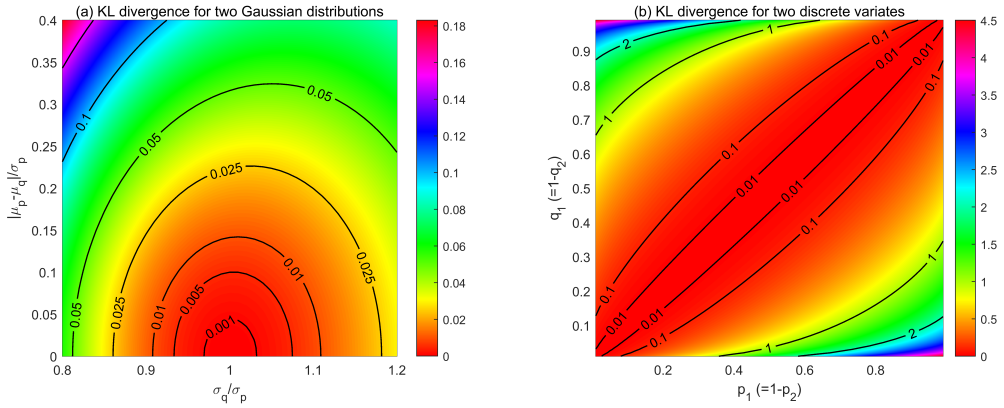


Figure 4.1: KL divergence: (a) two univariate normal distributions ($p(x) \sim \mathbb{N}(\mu_p, \sigma_p^2)$ and $q(x) \sim \mathbb{N}(\mu_q, \sigma_q^2)$), and (b) two discrete variate distributions ($p = [p_1, p_2]^T$ and $q = [q_1, q_2]^T$), where each black curved line corresponds to the same divergence value.

of our knowledge, this regularization strategy has so far applied to the objectives of reinforcement learning algorithms, which is an indirect way to control the exploration.

In reinforcement learning community, it is widely assumed that continuous actions will satisfy a Gaussian distribution with a diagonal covariance matrix and that discrete actions will follow a discrete distribution law. Additionally, the KL divergence is convex with respect to its two distributions [7], as shown in Figure 4.1. In such cases, many distinct distributions could induce the same KL divergence values. But, to encourage exploration [64], we have to search for some different policies, rather than simply using the approximated ones.

Our suggested alternative is to add an entropy regularization to the KL divergence constraint, to build an improved TRPO with entropy regularization for KL divergence. We term this method ERC-TRPO, which directly controls and adjusts the exploration of the agent using its constraint. Further, considering the viewpoint in Engstrom et al. [37], which highlights that code level optimization has a big impact on reinforcement learning, we also have provided an extremely detailed solution procedure for our ERC-TRPO, rather than a draft framework for solving the original TRPO [106]. Additionally, we also show that three TRPO-type methods have an almost equal time complexity theoretically. Finally, through an extensive series of experiments comparing ERC-TRPO with TRPO and ERO-TRPO in eight different benchmark environments, we can verify that ERC-TRPO is an effective and efficient reinforcement learning approach.

In summary, our work makes three contributions to this chapter:

- We propose a novel entropy regularized TRPO that adds Shannon entropy to the KL divergence constraint which prompts TRPO into better and wider explorations. To the best of our knowledge, this is the first TRPO variant to regularize its constraint to control the exploration directly;
- We provide a detailed solution procedure for our ERC-TRPO, and the same procedure for ERO-TRPO and TRPO, which benefit a fair experimental comparison for such three algorithms;
- Extensive experiments conducted within eight benchmark environments show just how effective and efficient ERC-TRPO is, via comparing to TRPO and ERO-TRPO.

The rest of this chapter is organized as follows. In Sections 4.2 and 4.3, we provide a mathematical description and detailed solution for our novel ERC-TRPO, respectively. We compare three TRPO-type methods in Section 4.4. Our detailed experiments are reported in Section 4.5. Finally, we summarize our work in this chapter.

4.2 TRPO with Entropy Regularization for KL Divergence Constraint

The aforementioned TRPO in Chapter 2 is either constrained in a trust region or strictly excludes all suspicious gradients, which suppresses exploration ability of the agent and thus hurts performance of the agent [71]. In this section, we apply Shannon entropy to improve the original TRPO, and then propose an enhanced TRPO with entropy regularization for the KL divergence constraint.

4.2.1 Shannon Entropy

Before deriving our novel TRPO variant, we introduce Shannon entropy and its special case for a Gaussian distribution [29].

Given a continuous random vector x, whose probability distribution is $p(x)$, its Shannon information entropy is defined as

$$H(x) = -\int p(x)\ln p(x)dx. \tag{4.1}$$

More specially, for a univariate normal distribution $p(x) \sim N(\mu, \sigma^2)$, we have

$$H(x) = \frac{1}{2}\big(\ln(2\pi e\sigma^2)\big). \tag{4.2}$$

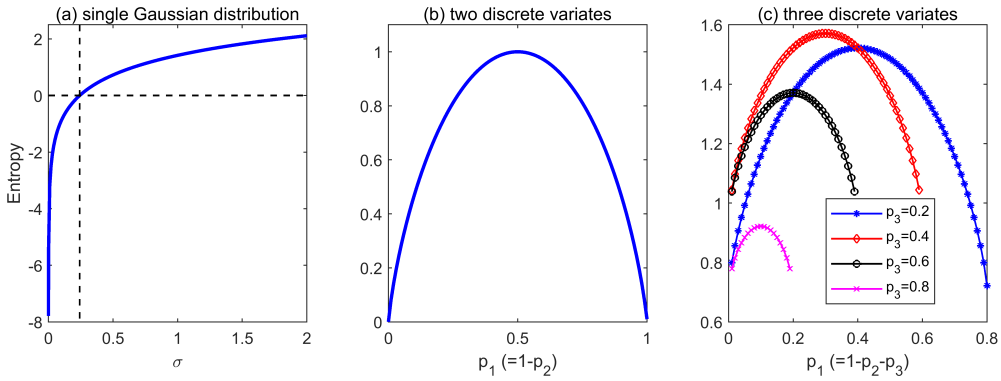


Figure 4.2: Shannon entropy for (a) an univariate normal distribution, (b) two discrete variates, and (c) three discrete variates.

This relation shows that the entropy $H(x)$ only depends on the variance σ^2. In addition, this entropy is possibly negative for $0 < \sigma < 0.242$. Further, when a n-variant random vector $x = [x_1, ..., x_i, ..., x_n]^T$ satisfies a normal distribution with a mean vector $\mu = [\mu_1, \mu_2, ..., \mu_n]^T$ and a diagonal covariance matrix (i.e., $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_n^2)$, the entropy becomes

$$H(x) = \sum_{i=1}^{n} \frac{1}{2}\left(\ln(2\pi e\sigma_i^2)\right). \tag{4.3}$$

It is worth noting that Eq. (4.3) is also additive, just like KL divergence in Chapter 2. In Figure 4.2 (a), we show the Shannon entropy curve for an univariate normal distribution, which is a monotone increasing function.

For a discrete distribution with n probability values, i.e., $p = [p_1, ..., p_i, ..., p_n]^T$ and $\sum_{i=1}^{n} p_i = 1$, the Shannon entropy is defined as

$$H(p) = -\sum_{i=1}^{n} p_i ln(p_i). \tag{4.4}$$

Figure 4.2 (b) and (c) illustrate the Shannon entropy for two and three discrete variates. As shown, this entropy is mathematically concave.

These definitions show that the Shannon entropy is an uncertainty metric [29]. The larger the entropy is, the more uncertain the system becomes. When this entropy is regarded as a regularization term to apply to TRPO, we can control the exploration to collect a more diverse trajectory to train a better policy, which reversely benefits our trajectory collection.

4.2.2 TRPO with Entropy Regularization for KL Divergence Constraint

To the best of our knowledge, existing entropy-regularized TRPOs mainly add an entropy regularization term into their objective functions. As an example, in this thesis, we only list the following form from Roosraie and Ebadzadeh's work [103]:

$$\begin{aligned} &\max \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t + \alpha H(\pi_\theta(a_t|s_t)) \right] \\ &\text{s.t.} \quad \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t)) \leq \delta \end{aligned} \tag{4.5}$$

where α indicates the regularization constant. We refer to this form as ERO-TRPO, noting that it can be solved using the aforementioned solution for TRPO in Chapter 2. This ERO-TRPO increases the uncertainty of the policy indirectly via adding an entropy regularizer to the surrogate objective, to increase the difference of two adjacent policies to enhance the exploration ability of the agent.

Next, we will outline how to derive our novel entropy regularized TRPO.

First, by regularizing the KL divergence with Shannon entropy, the original TRPO is reformulated into the following constrained maximization problem:

$$\begin{aligned} &\max \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \\ &\text{s.t.} \quad \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t)) - \beta H(\pi_\theta(a_t|s_t))] \leq \delta \end{aligned} \tag{4.6}$$

where $\beta > 0$ is a regularization constant. Here the constraint can adjust the trust region automatically and adaptively via the entropy term, to control the exploration ability effectively. After we define

$$\begin{aligned} &\hat{R}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \\ &\hat{D}_{KL}(\theta) = \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))] \\ &\hat{H}(\theta) = \hat{\mathbb{E}}_t[H(\pi_\theta(a_t|s_t))] \end{aligned} \tag{4.7}$$

the above enhanced TRPO is simplified into

$$\begin{aligned} &\max \hat{R}(\theta) \\ &\text{s.t.} \quad \hat{D}_{KL}(\theta) - \beta \hat{H}(\theta) \leq \delta \end{aligned} \tag{4.8}$$

where the constraint relaxes the upper bound of trust region to promote the exploration ability. Finally, we refer to this form as an enhanced trust region policy optimization with entropy regularization for a KL divergence constraint, or simply ERC-TRPO.

4.3 The Solution to ERC-TRPO

This section provides an approximate solution to our ERC-TRPO.

On the one hand, similarly in the TRPO of Chapter 2, we apply a quadratic approximation for the KL divergence and a linear one for the return.

On the other hand, since the entropy is monotone increasing for continuous actions and concave for discrete actions, as shown in Figure 4.2, their expectations (or averages over time steps) hold the same mathematical property. Therefore, the entropy is approximated linearly:

$$\hat{H}(\theta) = \hat{H}(\theta_{old}) + d^T \nabla \hat{H}(\theta_{old}) = \hat{H}(\theta_{old}) + h^T d \tag{4.9}$$

where the descent direction d and the gradient vector h are estimated as

$$\begin{aligned} d &= \theta - \theta_{old} \\ h &= \nabla \hat{H}(\theta_{old}). \end{aligned} \tag{4.10}$$

In this case, we could similarly build a linear-quadratic minimization problem:

$$\begin{aligned} &\mathrm{m}in \ -g^T d \\ &\mathrm{s}.t. \ \ \tfrac{1}{2} d^T H d - \beta h^T d \le \delta \end{aligned} \tag{4.11}$$

where g and H are respectively given in the followings:

$$\begin{aligned} g &= \nabla \hat{R}(\theta_{old}) \\ H &= \nabla^2 \hat{D}_{KL}(\theta_{old}). \end{aligned} \tag{4.12}$$

The corresponding Lagrangian function is defined as

$$L(d) = -g^T d + \lambda \left(\tfrac{1}{2} d^T H d - \beta h^T d - \delta \right). \tag{4.13}$$

Letting the gradient of $L(d)$ be zeros with respect to d, we have a set of linear equations:

$$Hd = \frac{1}{\lambda} g + \beta h \tag{4.14}$$

and thus the descent direction formally becomes

$$\begin{aligned} d &= \tfrac{1}{\lambda} H^{-1} g + \beta H^{-1} h \\ &= \tfrac{1}{\lambda} u + \beta v \\ &= \eta u + \beta v \end{aligned} \tag{4.15}$$

where two search directions (u and v) satisfy the following two sets of linear equations:

$$\begin{aligned} Hu &= g \\ Hv &= h. \end{aligned} \tag{4.16}$$

These two sets of linear equations can be solved using conjugate gradient method listed in Chapter 2. In (4.15), the parameter $\eta = 1/\lambda$ acts as a step size.

Revisiting the constraint in (4.11), we have a maximum η_{max} of the step size η:

$$\eta_{max} = \sqrt{\frac{2\delta + \beta^2 v^T H v}{u^T H u}}.$$

Similarly in the original TRPO, a linear search is conducted to achieve an optimal shrinking index i^* for the step size η

$$\begin{aligned} &\max \hat{R}(\theta_{old} + d) \\ &\text{s.t. } \hat{D}_{KL}(\theta_{old} + d) - \beta \hat{H}(\theta_{old} + d) \leq \delta \\ &\quad d = \eta_0^i \eta_{max} u + \beta v \\ &\quad i = 0, \ldots, n \\ &\quad \eta_0 \in (0, 1) \end{aligned} \tag{4.17}$$

and then a final update formulation is built for the policy:

$$\theta = \theta_{old} + \left(\eta_0^{i^*} \sqrt{\frac{2\delta + \beta^2 v^T H v}{u^T H u}} u + \beta v \right) \tag{4.18}$$

which will be simplified into (2.30) when v is removed. Theoretically, this gradient-based solution converges to a local minimum according to nonlinear optimization theory [15]. On the other hand, in the detailed implementation, we do not update the actor when the current objective is not larger than the previous one. These two aspects can achieve a good convergence for our proposed method.

The ERC-TRPO is summarized in **Algorithm**-3, where the above solution is to optimize the actor policy network, while the critic value network is optimized via a standard deep learning technique.

4.4 Comparison for Three TRPO-like Methods

In this section, we will compare three TRPO-type methods from three points of view: mathematical formulation, solution procedure, and time complexity.

Three TRPO-type methods follow an almost identical flowchart: to collect an episodic trajectory with the current policy, which then is used to train the actor and critic network

Algorithm 3 Entropy regularized constraint TRPO or ERC-TRPO

INPUT
  A specific experimental environment
  An actor network architecture
  A critic network architecture and its learning rate
  A reward discount factor γ
  A step size shrinking factor η_0
  A generalized advantage estimation factor λ
  The number of episodes N
  A random seed
INITIALIZATION
  Initialize the actor and critic networks
  Set a random seed in the environment
PROCESS
  Execute the following procedure till reaching N
    Collect trajectory data by running the current policy π_θ
      in the environment
    Calculate the generalized advantage estimation $\hat{A}(s_t, a_t)$
      across the entire trajectory
    Update the actor policy network as follows
      Calculate the gradients for the surrogate objective and entropy: g and h
      Solve two sets of linear equations to generate two vectors: u and v
      Execute a linear search for the step size
      Update the weights of the actor network θ
    Update the critic value network by minimizing the squared temporal
      difference error using a standard deep learning approach:
        $\hat{E}_t\left[\frac{1}{2}(r_t + \gamma V(s_{t+1}) - V(s_t))^2\right]$
OUTPUT
  The trained actor and critic networks

to improve the policy and state value function. During their iterative optimization, these methods apply the same data collection way, and thus their exploitation tricks are the same and relatively fixed. To control the tradeoff between exploitation and exploration, how to design and implement a good exploration becomes a key issue.

TRPO applies a KL divergence constraint to restrict the trained policy and the previous policy within a limited trust region to improve policy optimization stability. Since the KL divergence is non-linear and asymmetric, this constraint can result in the trained policy being close to the previous one, which can compress the exploration ability of the agent. In ERO-TRPO, an entropy regularizer is added to the surrogate objective, which can increase the policy uncertainty to improve the policy to collect a more diverse trajectory. However, this idea is indirectly to control the exploration. In our ERC-TRPO,

Table 4.1: Comparison of three TRPO-style algorithms in solution procedures

Method	TRPO	ERO-TRPO	ERC-TRPO
Approximated equation	$Hd = g$	$Hd = g + \alpha h$	$Hu = g, Hv = h$
Search direction	$d = H^{-1}g$	$d = H^{-1}(g + \alpha h)$	$u = H^{-1}g, v = H^{-1}h$
Decent direction	ηd	ηd	$\eta u + \beta v$
Step size	$\eta \leq \left(\frac{2\delta}{d^T H d}\right)^{1/2}$	$\eta \leq \left(\frac{2\delta}{(g+\alpha h)^T H (g+\alpha h)}\right)^{1/2}$	$\eta \leq \left(\frac{2\delta + \beta^2 v^T H v}{u^T H u}\right)^{1/2}$

Table 4.2: Comparison of three TRPO-style algorithms in time complexity

Operation	TRPO	ERO-TRPO	ERC-TRPO
Computing the gradient h	$O(n)$	$O(n)$	$O(n)$
Computing the gradient g		$O(n)$	$O(n)$
Solving a set of linear equations	$O(mn)$	$O(mn)$	$O(mn)$ $O(mn)$
Overall	$O(mn)$	$O(mn)$	$O(mn)$

Note:
n is the number of variables
m is the number of iterations

when its constraint is reformulated as the following form:

$$\hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))] \leq \delta + \beta\hat{\mathbb{E}}_t[H(\pi_\theta(a_t|s_t))] \tag{4.19}$$

this indicates that the trust region between two adjacent policies is adjusted automatically and adaptively via the entropy regularizer and its coefficient and thus a good policy can be optimized to collect a more diverse trajectory.

On the other hand, the main difference among the three methods stems from their solution to their constrained maximization problems. Their concise and key solution procedures are summarized in Table 4.1. Each is different in three aspects.

(a) **The approximated equation**: both TRPO and ERO-TRPO solve a set of linear equations, where the right term in ERO-TRPO is a linear combination of g and h, whereas ERC-TRPO needs to deal with two sets of linear equations: one related to g and the other related to h.

(b) **The search and descent directions**: ERO-TRPO firstly combines g and h linearly, and then finds a search direction. ERC-TRPO follows two search directions and linearly combines a descent direction subsequently.

(c) **The step size**: Compared to TRPO, ERO-TRP changes the denominator of the maximal step size, and whether or not its magnitude is larger than that of TRPO is

not deterministic. In ERC-TRPO, the numerator is enlarged as a result of the positive definite matrix H, implying that our ERC-TRPO has a larger upper bound on step size.

Finally, we analyze the time complexity of the aforementioned solutions in Table 4.1, which is shown in Table 4.2. The time complexity for estimating the gradients (g and h) is $O(n)$ [23]. For a single hidden layer network, $n = (|\mathscr{S}|+1)n_h + |\mathscr{A}|(n_h+1)$ for discrete actions, and $n = (|\mathscr{S}|+1)n_h + 2|\mathscr{A}|(n_h+1)$ for continuous actions, including a bias term for each hidden and output layer neuron, where n_h indicates the number of hidden layer neurons. In Chapter 2, we listed a time complexity $O(mn)$ for the conjugate gradient method. We use a similar approach for step size search on three methods, whose time complexity is neglected in this Table 4.2. In this case, overall, three methods have an approximate time complexity $O(mn)$ and thus will cost an approximate computational time experimentally, which will be validated in the next section.

4.5 Experiments

The following experiments were designed to validate ERC-TRPO, by comparing its performance with its two rivals: ERO-TRPO and TRPO in eight benchmark environments: Cart Pole, Acrobot, Pendulum, Inverted Pendulum, Reacher, Hopper, Bipedal Walker and Walker2D summarized in Appendix.

4.5.1 Experimental Settings

As mentioned, we validate ERC-TRPO, via comparing it to ERO-TRPO and TRPO. The basic code for TRPO comes from [1], which is used to implement ERO-TRPO and our ERC-TRPO.

These three approaches all rely on an actor network and a critic network. Our architecture for all three had a single hidden layer of 128 neurons with the number of neurons in the input layers decided using the dimensionality of observation state space.

In the actor network, the number of neurons in the output layer for discrete actions was decided by the dimensionality of the action space, or continuous actions (mean and variance), as twice the dimensionality. In particular, the weights and bias were trained by solving one or two sets of linear equations using a conjugate gradient method.

[1] https://github.com/boyu-ai/Hans-on-RL

Table 4.3: Network architecture and hyper-parameter settings

Hyper-parameter	Setting
Shared	
number of hidden layers	1
number of hidden neurons	128
activation function	ReLU
learning rate for critic	10^{-2} or 10^{-3}
optimizer	Adam
discounted factor (γ)	0.98 (discrete) or 0.90 (continuous)
generalized function factor (λ)	0.95 (discrete) or 0.90 (continuous)
horizon length	500(discrete) or 200(continuous)
TRPO	
KL threshold (δ)	tuning
ERO-TRPO	
entropy regularizer (α)	tuning
ERC-TRPO	
entropy regularizer (β)	tuning

In the critic network, there was only one neuron in the output layer to estimate the state value function. Here, the learning rate was set to 10^{-2} or 10^{-3}, as shown in Table 4.3.

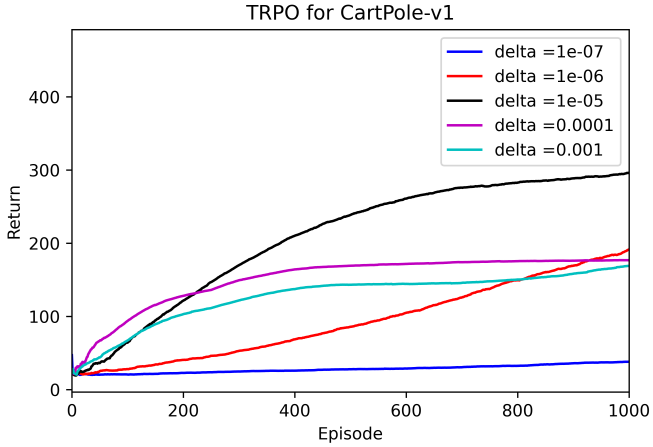
The discounted factor γ was 0.98 for the discrete action environments and 0.90 for the continuous action environments. λ for estimating the generalized advantage function was set to 0.95 for discrete action environments and 0.90 for continuous action environments. The linear search factor of step size was 0.5.

To speed up the training procedures for all three methods, we set the horizon length to 500 for two discrete action environments and 200 for six continuous action environments.

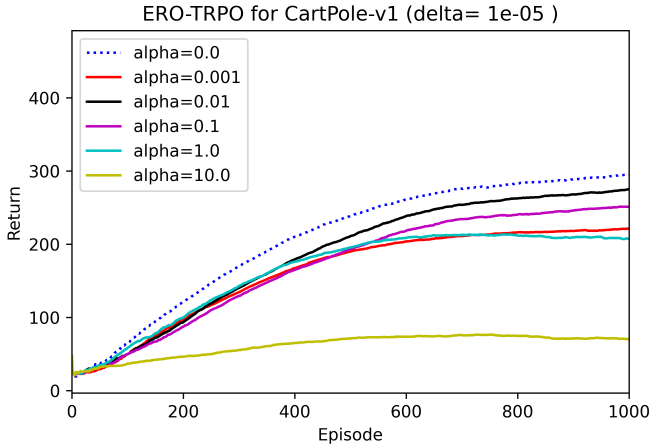
4.5.2 Hyper-parameter Tuning Procedure

TRPO has one hyper-parameter (δ), ERO-TRPO has two two parameters (δ and α), and REC-TRPO also has two parameters (δ and β).

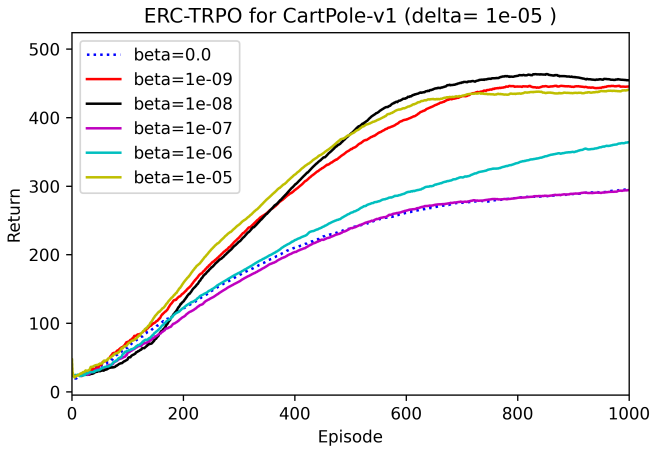
We tuned δ in TRPO among different values: $\delta \in \{10^{-1}, ..., 10^{-7}\}$. For the other two methods, we used a lazy strategy tuning $\alpha \in \{10^{-1}, ..., 10^{-9}\}$ and $\beta \in \{10^{-1}, ..., 10^{-9}\}$ only, given the optimal δ stayed fixed at the value from tuning TRPO.



(a) TRPO

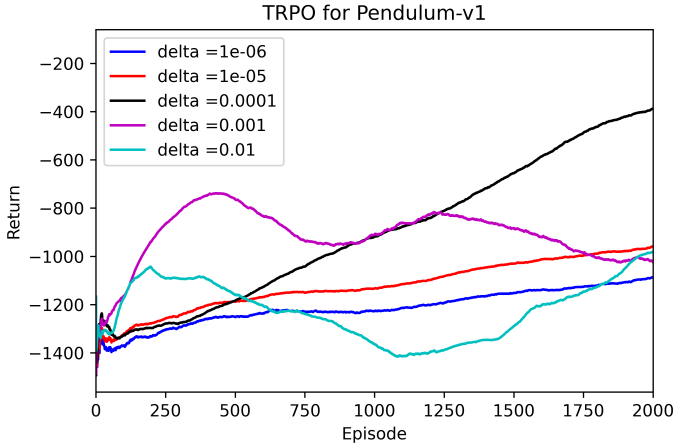


(b) ERO-TRPO

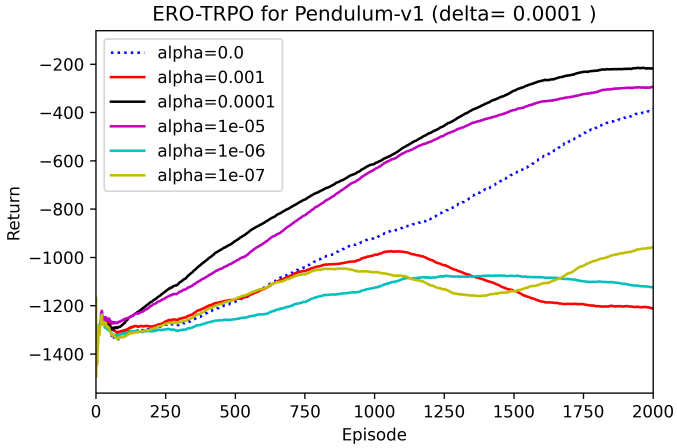


(c) ERC-TRPO

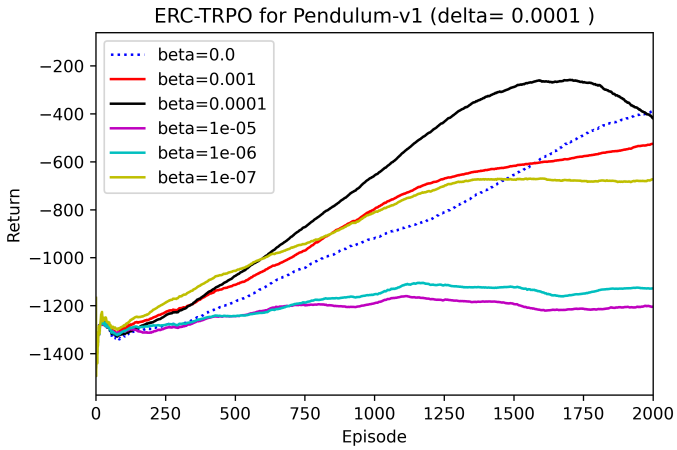
Figure 4.3: Parameter tuning for Cartpole environment with discrete actions.



(a) TRPO



(b) ERO-TRPO



(c) ERC-TRPO

Figure 4.4: Parameter tuning for Pendulum environment with continuous actions.

Table 4.4: Optimal hyper-parameters for different benchmark environments

Environment	Learning rate (critic)	TRPO δ	ERO-TRPO α	ERC-TRPO β
Cart Pole	10^{-2}	10^{-5}	10^{-2}	10^{-8}
Acrobat	10^{-2}	10^{-3}	10^{-1}	10^{-7}
Pendulum	10^{-2}	10^{-4}	10^{-4}	10^{-4}
Inverted Pendulum	10^{-3}	10^{-5}	10^{-4}	10^{-6}
Reacher	10^{-2}	10^{-2}	10^{-7}	10^{-6}
Hopper	10^{-2}	10^{-3}	10^{-4}	10^{-3}
Bipedal Walker	10^{-3}	10^{-3}	10^{-7}	10^{-7}
Walker2D	10^{-3}	10^{-2}	10^{-4}	10^{-4}

Table 4.5: Average returns over 5-6k episodes from three methods on eight environments

Environment	TRPO	ERO-TRPO	ERC-TRPO
Cart Pole	357.03(3)	395.45(2)	**414.18**(1)
Acrobot	-101.34(2)	-106.02(3)	**-99.33**(1)
Pendulum	-809.36(2)	-836.86(3)	**-599.77**(1)
Inverted Pendulum	57.46(3)	68.55(2)	**75.74**(1)
Reacher	-60.27(2)	-64.26(3)	**-45.23**(1)
Hopper	270.91(2)	216.72(3)	**313.99**(1)
BipedalWalker	-19.10(3)	-15.11(2)	**-9.54**(1)
Walker2d	182.63(3)	190.22(2)	**228.68**(1)
Mean	-15.26(2.5)	-18.91(2.5)	**34.84**(**1.0**)

Figures 4.5.2 and 4.5.2 show two detailed tuning procedures: the CartPole discrete action environment and the Pendulum single continuous action environment. It is worth noting that only five curves are plotted. The optimal curve is the middle value. The optimal hyper-parameters for the other benchmark environments are listed in Table 4.4.

In the above parameter tuning, the number of episodes to run is set to 1000 for discrete action environments or 2000 for continuous action environments, as shown in Figures 4.5.2 and 4.5.2, where the random seed number is fixed to 0.

Subsequently, to provide a solid comparison, we adjust the number of episodes to 6000. The results reported are the mean and standard deviation calculated over 10 trails based on 10 different random seed numbers.

4.5.3 Performance Evaluation of Sample Efficiency

The sample efficiency is a frequently-mentioned term, which is usually discussed in a qualitative way [34]. For example, [153] analyzes how and what impacts the sample efficiency in detail, but no quantitative metric is introduced.

In [82], besides the above generalization performance, it is also suggested to apply a sample efficiency for Procgen competition, which is also based on the normalized return calculation. Instead, we define two new metrics for sample efficiency, which are easier to understand and calculate than that in [82].

In classification performance evaluation, a receiver operating characteristic (ROC) is widely used [19, 40, 117]. And especially, the area under this ROC curve, named as AUC, ranges between 0.5 and 1.0, which can measure all decision thresholds including unrealistic ones.

In this thesis, this AUC is used to define an AUC-based metric to evaluate the sample efficiency:

$$SE_{auc} = AUC(ROC = \text{the curve of return}) \in (0,1]. \tag{4.20}$$

Assume that a return curve of size n is $R = \{R_1, ..., R_i, ..., R_n\}$ with an identical spacing, and the minimum and maximum of all compared cures are R_{min} and R_{max}. In this case, we can simplify the computation for SE_{auc}:

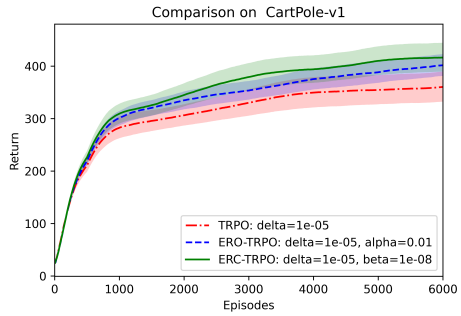
$$\begin{aligned} SE_{auc} &= \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{1}{2} \left(\frac{R_i - R_{min}}{R_{max} - R_{min}} + \frac{R_{i+1} - R_{min}}{R_{max} - R_{min}} \right) \\ &= \frac{1}{n-1} \frac{1}{R_{max} - R_{min}} \sum_{i=1}^{n-1} \frac{1}{2} (R_i + R_{i+1} - 2R_{min}) \\ &= \frac{1}{n-1} \frac{1}{R_{max} - R_{min}} \left(\sum_{i=1}^{n} R_i - \frac{1}{2}(R_1 + R_n) - (n-1)R_{min} \right). \end{aligned} \tag{4.21}$$

This is different from the normalized return based sample efficiency in [82]. Our metric measures an accumulative sample efficiency overall episodes or time steps.

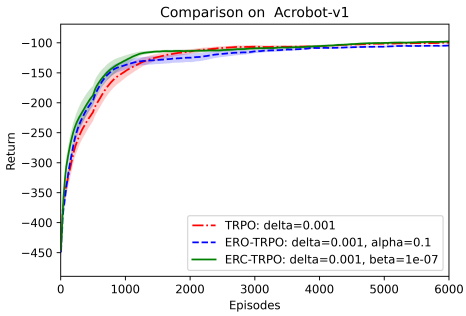
4.5.4 Experimental Results and Analysis

Figure 4.5 shows the results from our three-way comparison in a pictorial way.

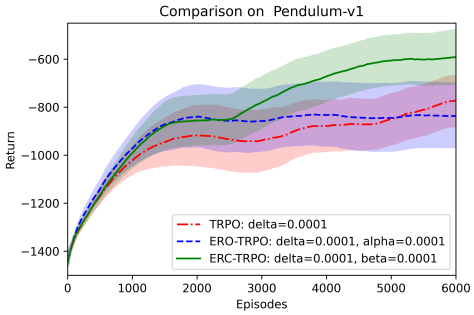
In terms of the two discrete action environments, ERC-TRPO worked slightly better than TRPO and ERO-TRPO in the Acrobot environment, but the difference is very slight. The results in the Cart Pole environment were much more clear cut, with ERC-TRPO outperforming the other methods by a large margin.



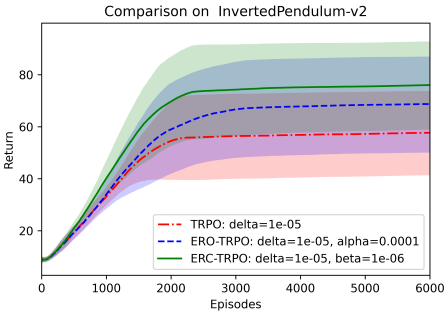
(a) Cart pole



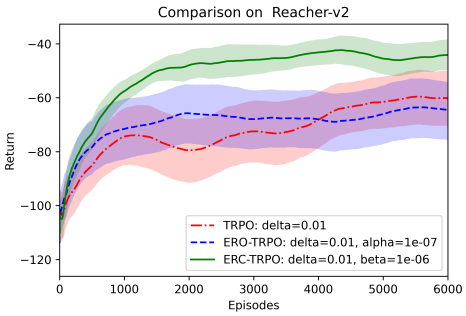
(b) Acrobot



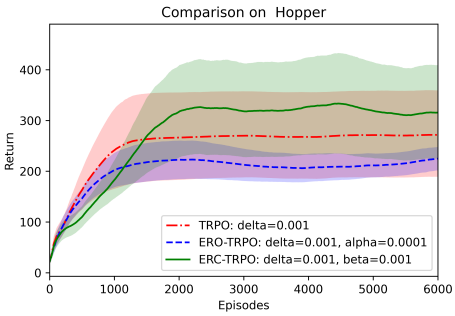
(c) Pendulum



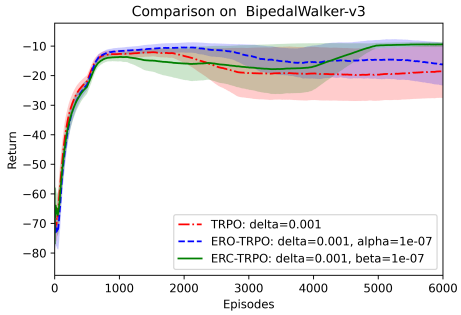
(d) Inverted pendulum



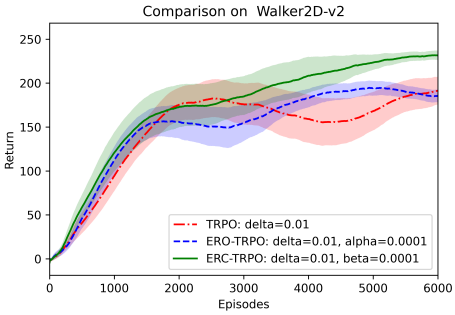
(e) Reacher



(f) Hopper



(g) BipedalWalker



(h) Walker2d

Figure 4.5: Performance comparison for three TRPO-type methods on eight benchmark environments

Table 4.6: Average computational time (second) to execute 6000 episodes

Environment	TRPO	ERO-TRPO	ERC-TRPO
Cart Pole	**1160.29**(1)	1287.53(3)	1242.02(2)
Acrobot	983.38(2)	973.71(3)	**894.29**(1)
Pendulum	1318.69(2)	1342.32(3)	**1227.97**(1)
Inverted Pendulum	**465.06**(1)	501.71(2)	528.40(3)
Reacher	694.63(2)	723.77(3)	**668.85**(1)
Hopper	1770.70(2)	**1462.58**(1)	2035.47(3)
Bipedal Walker	**2351.26**(1)	2397.07(2)	2422.94(3)
Walker2d	**2018.47**(1)	2047.42(2)	2218.87(3)
Mean	1345.31(**1.5**)	**1342.01**(2.38)	1404.85(2.12)

Turning to the six continuous action environments, ERC-TRPO yielded consistently superior results to TRPO and ERO-TRPO. The only exception was that ERO-TRPO performed worse than TRPO in the Hopper environment.

Moreover, we first calculate the average return over the last 1000 episodes, (i.e., from 5000 to 6000 episodes), and then estimate the average value across 10 trials, as shown in Table 4.5, where the best performance is highlighted in the **bold font**. According to [33], for each environment, the ranks among three methods are given in the bracket, where the small number indicates the better performance. It was found that our ERC-TRPO achieves the best performance. At the last raw named Mean, the overall average returns and average ranks over eight environments are provided, which also shows that our ERC-TRPO performs the best. Since TRPO is almost equal to ERO-TRPO using the mean rank and is superior to ERO-TRPO via the mean return, we claim that these two methods have an almost close performance.

Similarly, the computational time (second) is listed in Table 4.6, which demonstrates that, according to the mean rank of the last row, TRPO runs faster than our ERC-TRPO, and ERO-TRPO is the slowest, but on the basis of the meantime, our ERC-TRPO costs more computational time since our ERC-TRPO spends much more time than the other two methods on Hopper environment. Overall, the difference in computational time is very small.

Further, we estimate the sample efficiency based on AUC defined above and show its values from three methods on eight environments in Table 4.7. Except for Bipedal Walker, the highest sample efficiency values come from our ERC-TRPO. However, the difference between two entropy regularized methods is very small. In the last rows of Table 4.7, the mean over eight environments for each approach is shown, in which our

Table 4.7: AUC based sample efficiency from three techniques on eight environments.

Environment	TRPO	ERO-TRPO	ERC-TRPO
Cart Pole	0.7286	0.7927	**0.8321**
Acrobot	0.9195	0.9021	**0.9353**
Pendulum	0.5901	0.6373	**0.7443**
Inverted Pendulum	0.5964	0.7026	**0.8077**
Reacher	0.5809	0.6225	**0.8781**
Hopper	0.7286	0.5763	**0.8180**
Bipedal Walker	0.8600	**0.9047**	0.8944
Walker2D	0.6335	0.6506	**0.7458**
Mean	0.7047	0.7236	**0.8320**

ERC-TRPO is still the most efficient.

In short, these experimental comparisons empirically verify that ERC-TRPO is a highly effective and efficient technique for deep reinforcement learning.

4.6 Summary

In this chapter, we improve trust region policy optimization via a Shannon entropy regularization to its KL divergence constraint, to propose a novel entropy regularized trust region policy optimization, to enhance the exploration ability of trust region policy optimization. The superiority of the proposed method is demonstrated by some detailed experiments on eight benchmark environments, compared with two rivals: primary trust region policy optimization and its objective regularized version.

CHAPTER 5

TWIN TRUST REGION POLICY OPTIMIZATION

In this chapter, we will propose a twin trust region policy optimization, i.e., a novel extension of trust region policy optimization, which has an upper bound and a lower bound for its step size research.

5.1 Introduction

The original TRPO is formulated as a non-linear constrained optimization, as stated in Chapter 2. This formulation only provides an upper bound for the trust region. On the other hand, the TRPO currently is solved approximately and iteratively through four steps:

- The objective is expanded linearly and the constraint is expanded quadratically. Thus, what is originally a nonlinear constrained optimization problem is converted into a linear-quadratic programming problem.
- A set of linear equations for search direction is derived using a Lagrangian multiplier technique.
- A search direction is obtained via a conjugated gradient method.
- A maximal step size is estimated based on the limited KL divergence constraint, and proper step size is determined via a linear search.

From an optimization viewpoint [15, 120], a successful iterative solution depends on two factors: the search direction and the step size. However, the above procedural solution for TRPO does not include a lower bound for the step size search, which has a dramatic impact on evaluating the step size. Naturally, a lower bound can be defined manually, but it would not have a deterministic meaning. In this chapter, we aim to solve this issue, i.e., to bound the trust region from below for its optimization problem.

Our solution involves using a reciprocal optimization technique [39] to exchange the objective and the constraint in TRPO and minimize the KL divergence. This is, of course, subject to the constraint that the surrogate objective is greater than a predefined threshold. Via a similar approximation procedure in TRPO, we derive a quadratic programming problem. Then using the Lagrangian multiplier technique, we can obtain the same set of linear equations as that in TRPO for the search direction, and a lower bound for the linear search of the step size. We refer to this TRPO variant as reciprocal TRPO, or simply rTRPO.

In terms of the step size interval, TRPO provides an upper bound, while rTRPO gives a lower bound. Hence, we aggregate TRPO with rTRPO to build a novel TRPO, called twinTRPO, which effectively limits the step size search range. In this case, we also bound the trust region both from above and from below effectively.

To explore the efficacy of twinTRPO, we tested it on nine different benchmark environments, including continuous state and action spaces from classic control and MuJoCo: Mountain Car, Inverted Pendulum, Swimmer, Reacher, Half Cheetah, Walker2D, Pusher, Ant and Humanoid, which are arranged according to their number of actions in an ascending order. We also tested four different comparison methods: TRPO [106], PPO [107] and EnTRPO [103]), as well as our rTRPO.

We summarize our contributions in this chapter as follows:

- We propose a novel trust region policy optimization: rTRPO, based on the reciprocal optimization technique, which minimizes the KL divergence, subject to a least surrogate objective constraint, via exchanging the surrogate objective and the KL divergence constraint in the primary TRPO. This rTRPO bounds the trust region from below and thus induces a lower bound for the linear search of the step size.

- We aggregate TRPO and rTRPO to construct a novel TRPO: twinTRPO, which has both an upper bound and a lower bound for the linear search of the step size. This effectively limits the range of step size for policy optimization.

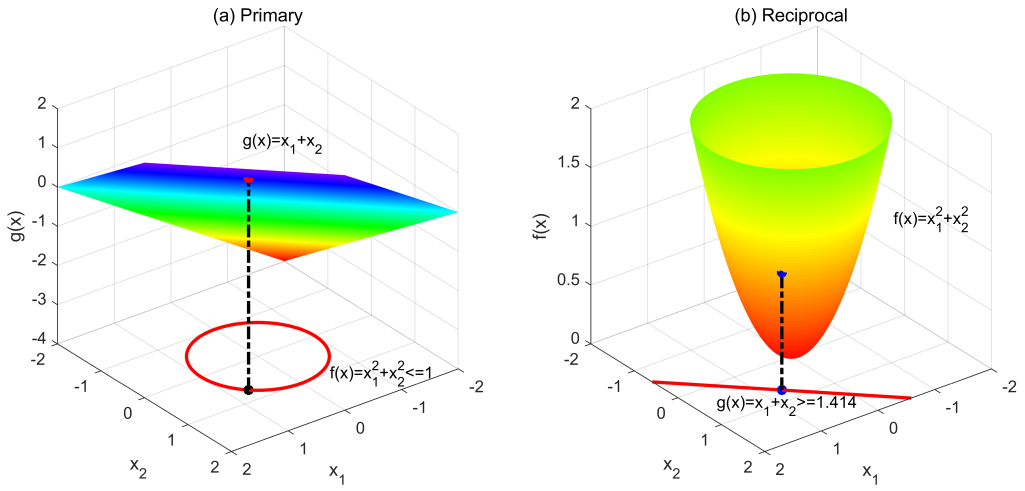


Figure 5.1: (a) Primary maximization problem and (b) reciprocal minimization problem.

- We validate the effectiveness and efficiency of twinTRPO, through conducting extensive experiments on nine benchmark reinforcement learning environments and comparing twinTRPO with three existing reinforcement learning methods and our rTRPO.

This chapter is organized as follows. Section 5.2 introduces reciprocal optimization technique. We propose our reciprocal trust region policy optimization and its solution, in Sections 5.3 and 5.4. Our twin trust region policy optimization is built in Section 5.5. The experimental results are reported in Section 5.6. At last, a simple summary is provided.

5.2 Reciprocal Optimization Technique

In [39], Favati and Pappalardo investigate two optimization problems. The primary problem is formulated as

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad g(x) \geq \alpha,\ x \in \mathbb{R}^n \end{aligned} \tag{5.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}$, $\alpha \in \mathbb{R}^n$.

Given $\beta \in \mathbb{R}$, the reciprocal problem associated with (5.1) becomes

$$\begin{aligned} &\max g(x) \\ &\text{s.t.} \quad f(x) \leq \beta \\ &\qquad x \in \mathbb{R}^n \\ &\qquad \beta \in \mathbb{R}. \end{aligned} \tag{5.2}$$

Sufficient conditions are established in [39] to guarantee that the optimal solution to (5.1) is also optimal for (5.2), and vice-versa. Formally, this is expressed as follows:

if x^* is the optimal solution of (5.1) and (5.2), then $\alpha = g(x^*)$ and $\beta = f(x^*)$.

Here, we give a simple two-dimensional example, as shown in Fig. 5.1, where we exchange the maximization problem with the minimization problem to suit our rTRPO. The primary maximization task is

$$\begin{aligned} &\max g(x) = x_1 + x_2 \\ &\text{s.t.} \quad f(x) = x_1^2 + x_2^2 \leq 1 \end{aligned} \tag{5.3}$$

whose optimal solution is $x_1 = x_2 = \sqrt{2}/2$ and its objective $g(x) = \sqrt{2}$. Its corresponding reciprocal minimization problem becomes

$$\begin{aligned} &\min f(x) = x_1^2 + x_2^2 \\ &\text{s.t.} \quad g(x) = x_1 + x_2 \geq \sqrt{2}. \end{aligned} \tag{5.4}$$

In Figure 5.1, the maximization and minimization problems have an identical solution, i.e., $x_1 = x_2 = \sqrt{2}/2$.

5.3 Reciprocal Trust Region Policy Optimization

According to the reciprocal optimization technique above, the original TRPO:

$$\begin{aligned} &\max \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \\ &\text{s.t.} \quad \hat{\mathbb{E}}_t \left[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t)) \right] \leq \delta \end{aligned} \tag{5.5}$$

is reformulated into the following constrained minimization problem:

$$\begin{aligned} &\min \hat{\mathbb{E}}_t \left[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t)) \right] \\ &\text{s.t.} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \geq \gamma_0. \end{aligned} \tag{5.6}$$

Its concise form is:

$$\begin{aligned} &\min \hat{D}_{KL}(\theta) \\ &\text{s.t.} \quad \hat{R}(\theta) \geq \gamma_0. \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} \hat{D}_{KL}(\theta) &= \hat{E}_t \left[D_{KL}(\pi_\theta(a_t|s_t) || \pi_{\theta_{old}}(a_t|s_t)) \right] \\ \hat{R}(\theta) &= \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right]. \end{aligned} \tag{5.8}$$

This formulation means that we should be able to find the minimal policy update that achieves the least return γ_0, which implies that we bound the trust region from below adaptively. This version of TRPO is referred to as reciprocal TRPO, or simply rTRPO.

5.4 The Solution to rTRPO

In this section, we present the procedure for solving rTRPO. At first, the same approximation strategy as used for TRPO is applied, i.e., a quadratic approximation for $\hat{D}_{KL}(\theta)$ and a linear one for $\hat{R}(\theta)$. This induces a standard quadratic programming problem:

$$\begin{aligned} &\min \tfrac{1}{2} d^T H d \\ &\text{s.t.} \quad g^T d \geq \gamma \end{aligned} \tag{5.9}$$

where the descent direction d, gradient vector g and Hessian matrix H are respectively defined as:

$$\begin{aligned} d &= \theta - \theta_{old} \\ g &= \nabla \hat{R}(\theta_{old}) \\ H &= \nabla^2 \hat{D}_{KL}(\theta_{old}). \end{aligned} \tag{5.10}$$

It is worth noting that the $\gamma > 0$ is a return increment from the previous policy to guarantee the return improvement. In addition, since the Fisher information matrix H is positive definite, this problem is a convex quadratic programming problem and thus has a unique solution.

Now we define a Lagrangian function to convert the problem (5.9) into an unconstrained problem:

$$L(d) = \frac{1}{2} d^T H d - \lambda (g^T d - \gamma) \tag{5.11}$$

where $\lambda > 0$ is a non-negative Lagrangian multiplier. Let the gradient of $L(d)$ with respect to d be zero, and we have a set of linear equations:

$$H d = \lambda g. \tag{5.12}$$

Let the step size $\eta = \lambda$, and we have a search direction u:

$$u = H^{-1} g. \tag{5.13}$$

as well as a descent direction:

$$d = \eta u. \tag{5.14}$$

It is worth noting that Eq.(5.13) is the same as Eq.(2.26) in Chapter 2, and can therefore also be solved via the conjugate gradient method [42]: **Algorithm 1** in Chapter 2.

Now, consider the constraint in (5.9):

$$g^T d = \frac{1}{\eta} d^T H d = \eta u^T H u \geq \gamma. \tag{5.15}$$

We have a minimal step size:

$$\eta_{min} = \frac{\gamma}{u^T H u} > 0. \tag{5.16}$$

Since the Fisher information matrix H is positive definite, the denominator is positive. To guarantee a positive η_{min}, we would set a positive γ:

$$\gamma = \rho |\hat{R}(\theta_{old})| \tag{5.17}$$

and thus we have

$$\gamma_0 = \hat{R}(\theta_{old}) + \rho |\hat{R}(\theta_{old})| \tag{5.18}$$

where $\rho \in \{0, 1\}$ is a factor to control the return improvement.

Moreover, since the original $\hat{R}(\theta)$ is nonlinear, to ensure a maximal improvement of the return $\hat{R}(\theta)$, we estimate a maximal KL divergence with the minimal step size η_{min}:

$$\hat{D}_{KL}^{max} = \hat{D}_{KL}(\theta_{old} + \eta_{min} u) \tag{5.19}$$

and search for an expanding index i^* to maximize the improvement of return. In this case, the following linear search procedure applies:

$$\begin{aligned} &\max \hat{R}(\theta_{old} + \eta_e^i \eta_{min} u) \\ &\text{s.t. } \hat{D}_{KL}(\theta_{old} + \eta_e^i \eta_{min} u) \leq \hat{D}_{KL}^{max} \\ &\quad i = 0, ..., n \\ &\quad \eta_e \geq 1 \end{aligned} \tag{5.20}$$

where n is the number of linear searches, e.g., $n = 15$, and $\eta_e > 1$ is an expanding factor. Lastly, we update the θ vector:

$$\theta = \theta_{old} + \eta_e^{i^*} \frac{\rho \hat{R}(\theta_{old})}{u^T H u} u \tag{5.21}$$

which is used to train the actor in our rTRPO.

Algorithm 4 Twin trust region policy optimization (twinTRPO)

INPUT
  A specific experimented environment
  An actor network architecture, and a critic network architecture, and a learning rate
  A reward discount factor γ, two step size shrinking and expanding factors η_s and η_e
  A generalized advantage estimation factor λ
  The number of episodes N and a random seed
INITIALIZATION
  Initialize the actor and critic networks
  Set the random seed for the environment
PROCESS
  Execute the following procedure till reaching N
    Collect trajectory data by running the current policy π_θ in the environment
    Calculate the generalized advantage estimation $\hat{A}(s_t, a_t)$ across the entire trajectory
    Update the actor policy network as follows
      Calculate the gradients for the surrogate objective: g
      Estimate the Fisher information matrix for KL divergence: H
      Solve a set of linear equations to achieve the search direction: u
      Estimate the upper and lower bounds of the step size (η_{min} and η_{max})
      Execute a linear search to achieve an optimal step size η^*
      Update the weights of the actor network θ
    Update the critic value network via minimizing the squared temporal difference error:
      $\hat{E}_t\left[\frac{1}{2}(r_t + \gamma V(s_{t+1}) - V(s_t))^2\right]$
OUTPUT
  The trained actor and critic networks

5.5 Twin Trust Region Policy Optimization

From the TRPO in Chapter 2 and the above rTRPO algorithms, one can observe that:

(a) Both TRPO and rTRPO solve the same set of linear equations for their search direction: $Hu = g$.

(b) rTRPO provides the minimal step size (η_{min}), while TRPO gives the maximal step size (η_{max}), i.e.,

$$\begin{aligned} \eta_{min} &= \frac{\rho|\hat{R}(\theta_{old})|}{u^T H u} \\ \eta_{max} &= \left(\frac{2\delta}{u^T H u}\right)^{\frac{1}{2}}. \end{aligned} \tag{5.22}$$

In this section, we aggregate TRPO with rTRPO to build twin TRPO, in which the

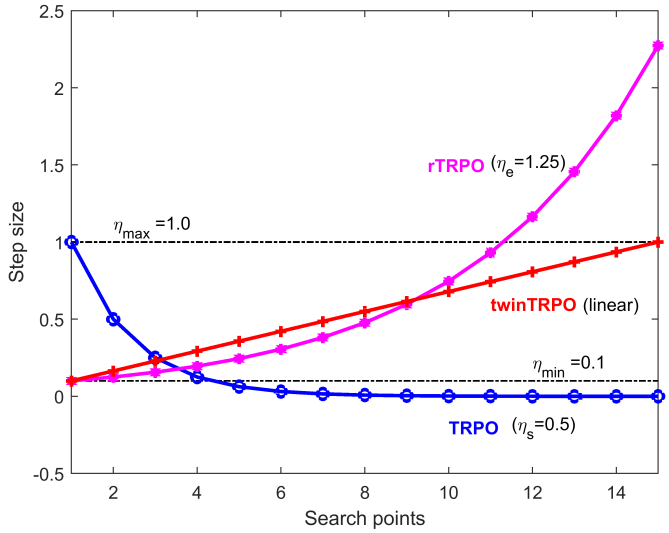


Figure 5.2: Step size search settings for TRPO, rTRPO and twinTRPO.

η_{min} and η_{max} are regarded as the lower and upper bounds for step size search. There are two cases to be analyzed:

(a) $\eta_{min} \leq \eta_{max}$. These two values are regarded as the lower and upper bounds of the step size, from which an optimal step size η^* can be determined, such that

$$\begin{aligned} &\max \hat{R}(\theta_{old} + \eta_i u) \\ &\text{s.t.} \quad \hat{D}_{KL}(\theta_{old} + \eta_i u) \leq \delta \\ &\qquad \eta_i = \eta_{min} + i\frac{\eta_{max} - \eta_{min}}{n-1} \\ &\qquad i = 0, \ldots, n-1. \end{aligned} \tag{5.23}$$

Notably, this is implemented using a pure linear search. Figure 5.2 shows three different step size search settings for TRPO, rTRPO, and twinTRPO. As the search index increases, the step size for TRPO tends to be stable, and the step size for rTRPO increases exponentially. This can induce difficulty in optimal step size detection.

(b) $\eta_{min} > \eta_{max}$. In this case, only these two bounds are checked to detect which one is better:

$$\begin{aligned} &\max \hat{R}(\theta_{old} + \eta_i u) \\ &\text{s.t.} \quad \eta_i = \eta_{min} \text{ or } \eta_{max}. \end{aligned} \tag{5.24}$$

The **Algorithm 4** summarizes the pseudo-code for twinTRPO, detailing how the solution provides both an upper bound and a lower bound for the step size search. Obviously, this helps to substantially improve the optimization procedure.

Table 5.1: Key parameter settings for five algorithms

Parameter	Settings
Shared parameters for all five algorithms	
Discount factor	0.9
GAE parameter	0.9
KL constraint	5.0×10^{-5}
Fixed parameters for different algorithms	
Learning rate of actor for PPO	10^{-5}
Clipping coefficient for PPO	0.2
Return increment factor for rTRPO and twinTRPO	10^{-3}
Step size shrinking factor for TRPO, PPO, EnTRPO	0.5
Step size expanding factor for rTRPO	1.25
Regularization coefficient for EnTRPO	10^{-4}
Tuned parameters for different algorithms	
Neurons of hidden layers for TRPO and EnTRPO	64
for PPO, rTRPO and twinTRPO	128
Learning rate of critic	
for TRPO, EnTRPO and PPO	5×10^{-3}
for rTRPO and twinTRPO	10^{-2}

5.6 Experiments

This section presents the experiments we undertook to validate twinTRPO, against TRPO [106], PPO [107], EnTRPO [103], and our rTRPO, on nine benchmark environments: Mountain Car (continuous), Inverted Pendulum, Swimmer, Reacher, Half Cheetah, Walker2D, Pusher, Ant and Humanoid, as stated in Appendix. It is worth that these environments are arranged according to their dimension of action space.

5.6.1 Experimental Settings

The basic TRPO and PPO software is downloaded at [1], which is adopted to implement EnTRPO, and our rTRPO and twinTRPO. There are several parameters in these five compared algorithms, as shown in Table 5.1. We divide these parameters into three groups: shared, fixed, and tuned.

As shown in Table 5.1, there are three parameters to be used in all five algorithms. The fixed parameters come from the default settings of some public software. We tuned

[1] https://github.com/boyu-ai/Hands-on-RL

Table 5.2: Mean and standard deviation from the last 1000 episodes

Environment	TRPO	PPO	EnTRPO	rTRPO	twinTRP(ours)
Mountain Car	-15.49±1.18	-2.65±0.80	-11.14±2.12	-5.15±0.37	**0.00**±0.00
Inverted Pendulum	914.72±24.65	**968.92**±15.62	910.28±23.20	197.16±73.56	917.05±18.94
Swimmer	29.61±0.30	31.95±0.33	16.44±1.48	39.06±0.16	**41.94**±0.29
Reacher	-29.18±0.76	-11.44±0.36	-29.25±0.56	-11.80 ±0.26	**-10.17**±0.25
Half Cheetah	534.65±73.13	265.04±72.69	526.92±69.72	91.91±40.26	**976.14**±25.35
Walker2D	146.53±5.66	97.77±3.35	228.13±6.30	165.41 ±13.21	**251.89**±2.18
Pusher	-72.85±0.54	-67.06±1.87	-74.17±0.58	-47.30 ±0.58	**-43.12**±0.45
Ant	1020.88±13.27	947.54±5.36	1006.72±12.24	977.85 ±6.94	**1115.24**±9.05
Humanoid	335.25±5.48	314.95±4.38	334.41±4.22	201.58 ±9.23	**337.57**±4.15

two aspects of parameters: the number of hidden layers, and the learning rate of critic, on Reacher environment, given a fixed random seed.

The actor and critic networks had the same network architecture with two hidden layers. We checked 64 and 128 neurons of hidden layers and then detected 64 for TRPO and EnTRPO, and 128 for PPO, rTRPO, and twinTRPO.

We tuned the learning rate of critic with four values: 5×10^{-2}, 10^{-2}, 5×10^{-3} and 10^{-3}. Then, we determined 5×10^{-3} for TRPO, PPO and EnTRPO, and 10^{-2} for rTRPO and twinTRPO.

The parameters in Table 5.1 were subsequently applied to eight remaining environments to verify five compared algorithms.

5.6.2 Experimental Results and Analysis

We conducted 10 trails using 10 different random seeds on nine benchmark environments and showed the results in Figure 5.3.

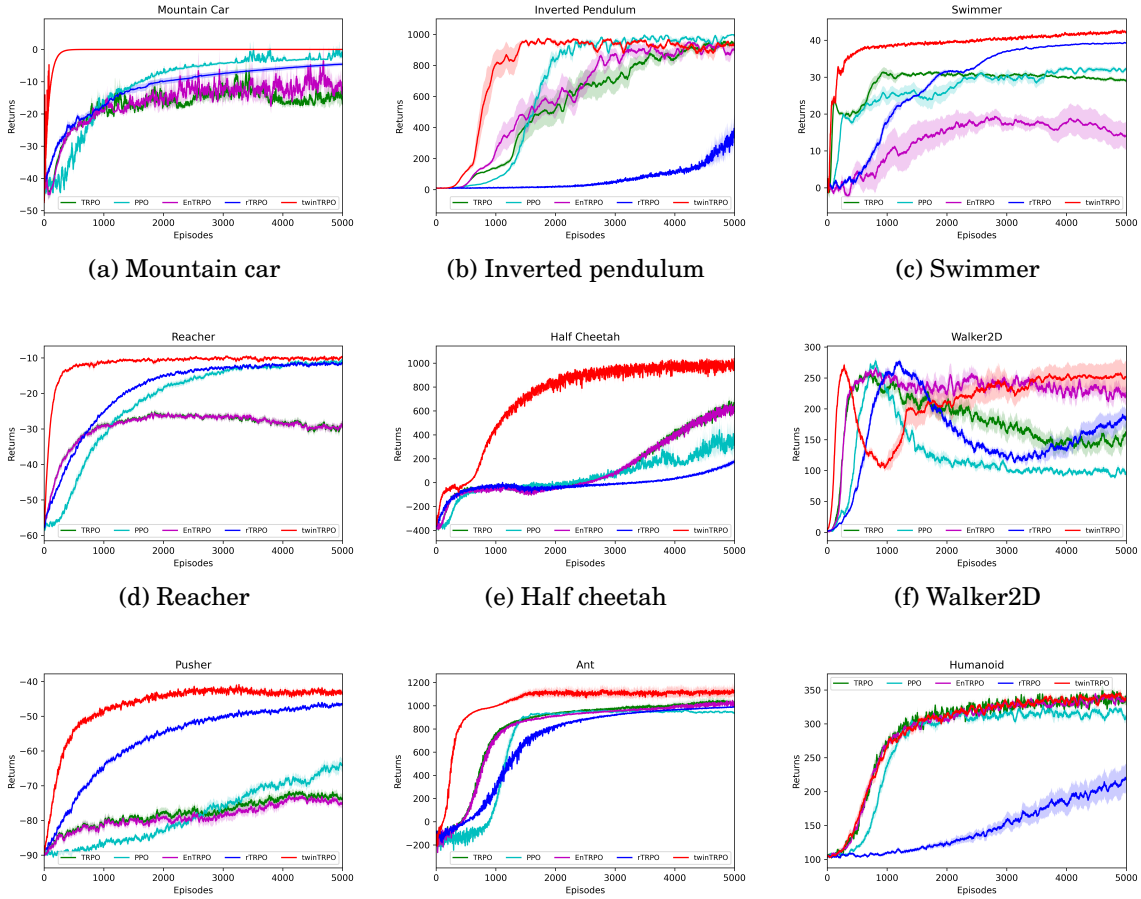
From the results, compared to TRPO, we can see that

(a) PPO works better than TRPO on five environments (i.e., Mountain Car, Inverted Pendulum, Swimmer, Reacher, and Pusher), and as well as TRPO on Ant.

(b) EnTRPO is superior to TRPO on Mountain Car and Walker2D, and is as good as the other seven environments but Swimmer.

(c) rTRPO was able to continuously increase rewards. Moreover, it performs better than TRPO on the other six environments excluding Inverted Pendulum, Half Cheetah, and Humanoid.

(d) Our twinTRPO increased its rewards very quickly and then tended to stabilize. Additionally, this method had an extremely small standard deviation. Overall, our



(a) Mountain car (b) Inverted pendulum (c) Swimmer

(d) Reacher (e) Half cheetah (f) Walker2D

(g) Pusher (h) Ant (i) Humanoid

Figure 5.3: Performance comparison on nine benchmark environments.

twinTRPO outperforms TRPO in the other eight environments, except for Inverted Pendulum.

Then, we estimated the mean and standard derivations of the returns during the last 1000 episodes (4000-5000), as shown in Table 5.2. Here, our twinTRPO achieves the best performance in eight environments and the second best on Inverted Pendulum where PPO works the best.

Lastly, we calculated the sample efficiency metric based on AUC defined in Chapter 4 for five RL methods on nine benchmark environments, as shown in Table 5.3. Our twinTRPO is the most efficient in all environments except for Humanoid, in which TRPO works more efficiently than our twinTRPO with an extremely small difference (0.0001). According to the last rows in Table 5.3, our twinTRPO has the highest sample efficiency

Table 5.3: AUC based sample efficiency of five RL methods on nine environments.

Environment	TRPO	PPO	EnTRPO	rTRPO	twinTRPO
Mountain Car	0.6291	0.7719	0.6576	0.7526	**0.9837**
Inverted Pendulum	0.5547	0.6556	0.6039	0.0596	**0.7877**
Swimmer	0.6902	0.6501	0.3465	0.6817	**0.9192**
Reacher	0.5918	0.7437	0.5908	0.8059	**0.9567**
Half Cheetah	0.3603	0.3101	0.3562	0.2656	**0.7741**
Walker2D	0.6368	0.4309	0.8128	0.5577	**0.7621**
Pusher	0.2564	0.2441	0.2328	0.6843	**0.8802**
Ant	0.7606	0.6908	0.7481	0.6811	**0.9136**
Humanoid	**0.7474**	0.6790	0.7403	0.1739	0.7473
Mean	0.5808	0.5751	0.5655	0.5181	**0.8583**

among five RL methods over nine environments.

In short, these experiments validate the superiority of twinTRPO, in both reinforcement learning performance and sample efficiency.

5.7 Summary

In this chapter, to remedy the lack of lower bound in the step size search of trust region policy optimization, we revisit a reciprocal optimization technique and then correspondingly build a reciprocal trust region policy optimization by exchanging the objective and the KL divergence constraint. This procedure yields a lower bound for the step size. Moreover, fusing trust region policy optimization and its reciprocal version into a framework we call twinTRPO provides a solution that gives an upper bound and a lower bound to effectively control policy searches. From experiments with nine benchmark environments and four comparison methods, we find twinTRPO to be a highly effective and efficient solution.

CHAPTER 6

IMPROVING PROXIMAL POLICY OPTIMIZATION WITH ALPHA DIVERGENCE

In this chapter, we will revisit the primary proximal policy optimization which is extended by a linear combination formulation and alpha divergence to enhance its performance.

6.1 Introduction

In the previous two chapters, two enhanced trust region policy optimization algorithms have been investigated. They need a second-order solution technique, that is, a Hessian matrix is evaluated at each episode update. Therefore, it is time-consuming to solve these non-linear constrained optimization problems.

In Chapter 3, we also review a famous variant of trust region policy optimization, i.e., proximal policy optimization (PPO). In PPO, the constrained optimization problem in TRPO is converted into an unconstrained optimization via the Lagrangian multiplier technique, introducing a penalty parameter, this will be called a primary PPO in this thesis. This PPO can be solved by a first-order solution approach, that is, a gradient-based optimization, at each episode update.

Moreover, two variants are derived: adaptive PPO and clipping PPO. The former adaptive form is to dynamically adjust the penalty parameter during the training procedure, in which three additional tunable parameters are added. This means that a

more complicated parameter tuning is needed. The latter clipping form discards the KL divergence, by applying a clipping function to limit the advantage function. So far, more improvements have been made to the clipping PPO, for example, to modify the clipping function forms [135, 163], to add policy feedback from actor to critic, and to check the trust region outside of policy updates [136].

Because the primary PPO only covers one tunable penalty parameter which is difficult to tune and then is outperformed by the clipping one, it has received less attention yet to date [107]. In the meantime, there exists a discordant fact that the clipping function is a rough and indirect difference between two policies while KL divergence is usually considered an effective difference description. In this chapter, we address such two problems to improve the effectiveness of this PPO version: (a) how to control the trade-off between two terms more conveniently, and (b) how to measure the difference between two adjacent policies, i.e., the trust region, more effectively for different environments.

Generally speaking, the magnitude of return is much greater than that of KL divergence, which makes it difficult for the tuning procedure of the penalty parameter to balance two terms. In this chapter, the original objective is converted into a linearly combined form (Combined PPO) via a balanced factor, which is at the interval (0,1). This novel form will adjust the balanced factor more conveniently.

As reviewed in Chapter 2, the original KL divergence is a parameter-free divergence, which is recognized as a relatively good ability to describe the difference between two policies. In order to characterize this difference in our method more elaborately, we introduce the alpha divergence, which has properties that it is continuous for the real variable α in the whole range including two singularities $\alpha = 0, 1$ and is convex with respect to both policies [27]. Furthermore, the alpha divergence is a generalized form of KL one when alpha tends to 1, and includes several existing divergences, for example, Pearson Chi-square divergence, Hellinger distance, reverse Pearson/Neyman Chi-square divergence and reverse KL divergence, when alpha is set to 2, 1/2, -1 and to 0, respectively [7].

Finally, a new PPO algorithm is built with a linear combined objective and alpha divergence, which is referred to simply as alphaPPO. We conduct experiments to verify the effectiveness of the presented alphaPPO method by comparing it with the clipping PPO and combined PPO on six benchmark environments.

In summary, there are two contributions of this chapter:

- After the original penalty objective is converted to a linearly combined form, we replace KL divergence using a parametric alpha divergence, to measure the difference

between two sequential policies;

- Our proposed alphaPPO is verified by extensive experiments on six benchmark environments in traditional control tasks and MuJoCo simulated environments.

This chapter is organized as follows. In Sections 6.2, we introduce PPO in detail. Our novel alphaPPO will be presented in Section 6.3. In Section 6.4, the result of some experiments will be shown. Finally, we will summarize our study in this chapter.

6.2 Proximal Policy Optimization

Proximal policy optimization (PPO) [107] stems from TRPO, so here we restate TRPO first [106]. TRPO is formulated as a constrained maximization problem:

$$\begin{aligned} &\max \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \\ &\text{s.t.} \quad \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))] \leq \delta \end{aligned} \tag{6.1}$$

where the objective function indicates the generalized advantage function defined in Chapter 2, θ and θ_{old} are the current and previous policies, respectively, $D_{KL}(\cdot)$ indicates the KL divergence, and δ is the threshold of KL divergence. In Chapter 2, we have reviewed and analyzed the KL divergence in detail.

PPO converts the above problem (6.1) into an unconstrained optimization problem:

$$\max \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] - \beta \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))]. \tag{6.2}$$

Let

$$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{6.3}$$

which is usually referred to as a probability ratio. Further, we define two terms in (6.2) as

$$\bar{R}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[\rho_t(\theta) \hat{A}_t \right] \tag{6.4}$$

and

$$\bar{D}(\theta) = \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))]. \tag{6.5}$$

In this case, we have a more concise representation for PPO

$$\max f(\theta) = \bar{R}(\theta) - \beta \bar{D}(\theta). \tag{6.6}$$

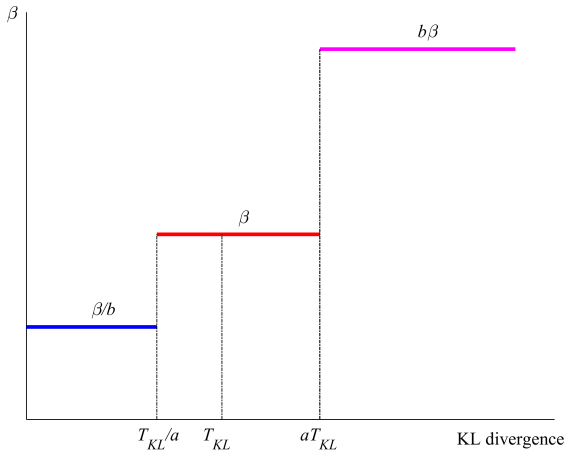


Figure 6.1: Dynamical update for β in adaptive PPO.

This PPO has been used to derive two more widely-cited variants. One is an adaptive version, as shown in Figure 6.1, which adjusts the β value dynamically, i.e.,

$$\beta = \begin{cases} \beta/b, \text{if}\, \bar{D}(\theta) \le T_{KL}/a \\ \beta b, \text{if}\, \bar{D}(\theta) \ge a T_{KL} \\ \beta, \text{otherwise} \end{cases} \tag{6.7}$$

where β is a current penalty factor setting, T_{KL} denotes a threshold for KL divergence, and a and b are two factors to enlarge and shrink the β value, respectively.

The other is a clipping version, which is formulated as

$$\max \hat{\mathbb{E}}_t \left[\min \left\{ \rho_t(\theta) \hat{A}_t, \text{clip} \left\{ \rho_t(\theta), 1-\varepsilon, 1+\varepsilon \right\} \hat{A}_t \right\} \right] \tag{6.8}$$

where ε is a threshold for the probability ratio $\rho_t(\theta)$, and the clip function is defined as

$$\text{clip}(x) = \begin{cases} 1+\varepsilon, \text{ if}\, x \ge 1+\varepsilon \\ 1-\varepsilon, \text{ if}\, x \le 1-\varepsilon \\ x, \text{ otherwise} \end{cases} \tag{6.9}$$

as shown in Figure 6.2.

Here, we analyze some possible limitations of the aforementioned three PPO forms. In the primary PPO (6.2), there is one tunable factor β only. However, it has been experimentally determined that the magnitude of $\bar{R}(\theta)$ is much larger than that of $\bar{D}(\theta)$,

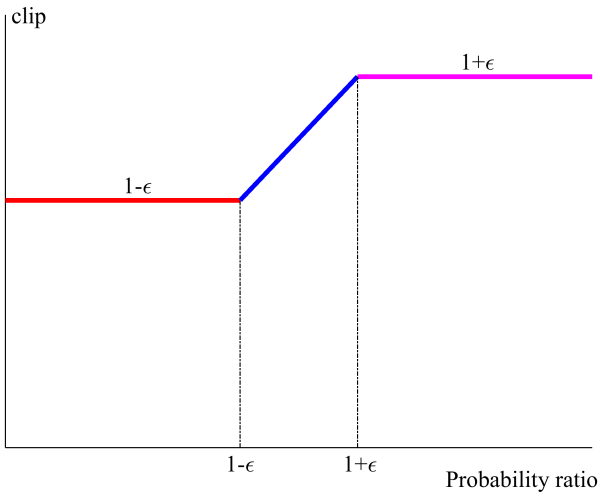


Figure 6.2: The clipping function used in clipping PPO.

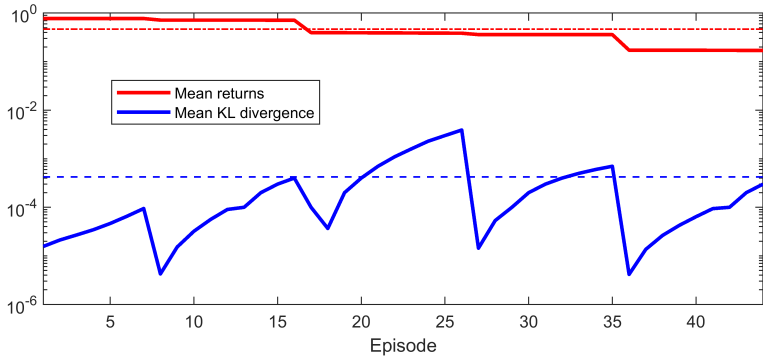


Figure 6.3: Mean return and KL divergence in each batch from Pendulum.

as shown in Figure 6.3. In this case, it is extremely difficult to tune a proper β value for real-world applications.

Besides an initial β, the adaptive PPO (6.7) introduces three new factors (a, b and T_{KL}), that would affect the performance of PPO greatly and thus would result in a complicated tuning procedure. In particular, when a large T_{KL} is chosen (e.g., 10^{-2}), the β value would be reduced gradually and then tends to zero, if all KL divergence values ($< 10^{-3}$) were less than T_{KL}. This situation could neglect the effect from KL divergence.

For the clipping PPO, it is widely criticized that the factor ε is not associated with the KL divergence directly [135], although it has been applied in many applications.

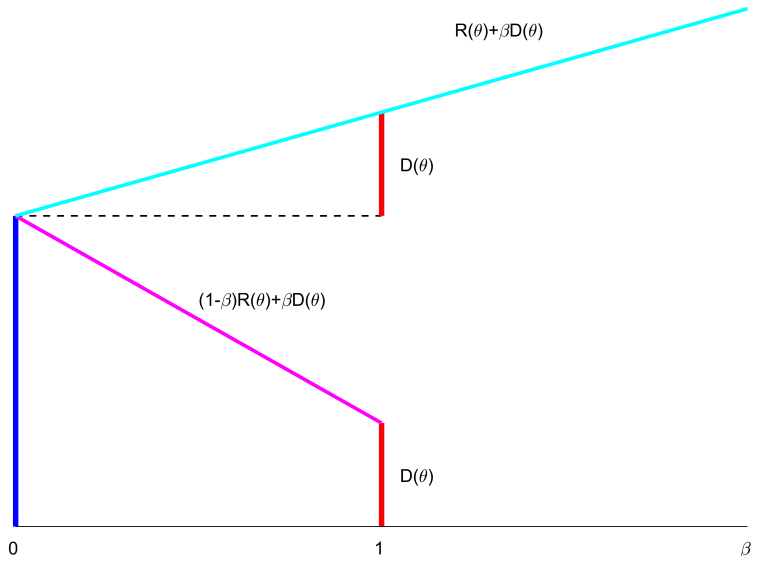


Figure 6.4: Two different objective forms in primary PPO and our PPO.

6.3 An Improved PPO with Alpha Divergence

In this study, we will improve the primary PPO in two aspects. One is to reformulate it as a linearly combined form and the other is to replace KL divergence with alpha divergence.

6.3.1 Linearly Combined Objective for Primary PPO

At first, we rewrite the above problem (6.6) as the following minimization problem using a linear combination:

$$\min f(\theta,\beta) = -(1-\beta)\bar{R}(\theta) + \beta\bar{D}(\theta) \tag{6.10}$$

where $\beta \in [0,1]$ is referred to as a balanced factor. The comparison of two different objective forms in primary PPO and our PPO is shown in Figure 6.4. Theoretically, this function could be rewritten as

$$\min f(\theta,\beta) = -\bar{R}(\theta) + \frac{\beta}{(1-\beta)}\bar{D}(\theta) \tag{6.11}$$

which looks like the objective in (6.6). However, the actor in PPO uses a fixed learning rate, and thus the optimization procedure for (6.10) is slightly different from that for (6.6). We refer to this form (6.10) as a combined PPO in this study.

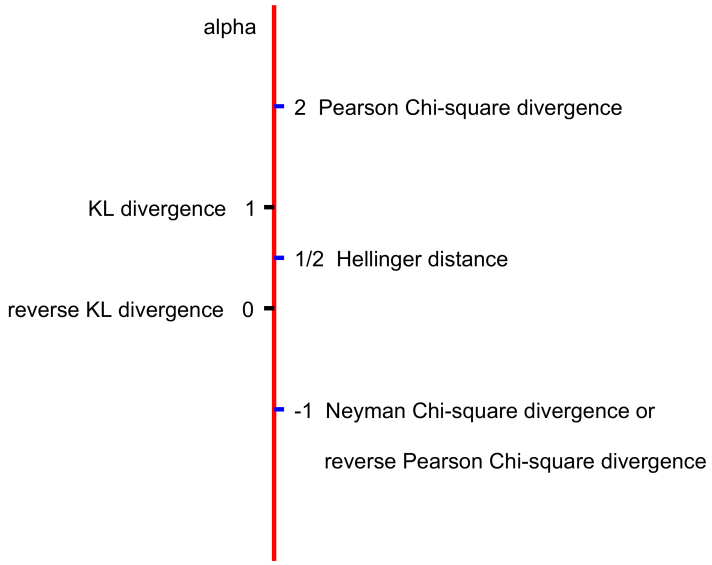


Figure 6.5: Five specific divergences with different alpha values.

6.3.2 Alpha Divergence

The aforementioned KL divergence is a parameter-free divergence. In order to measure two different distributions more accurately, we introduce an alpha divergence in this study.

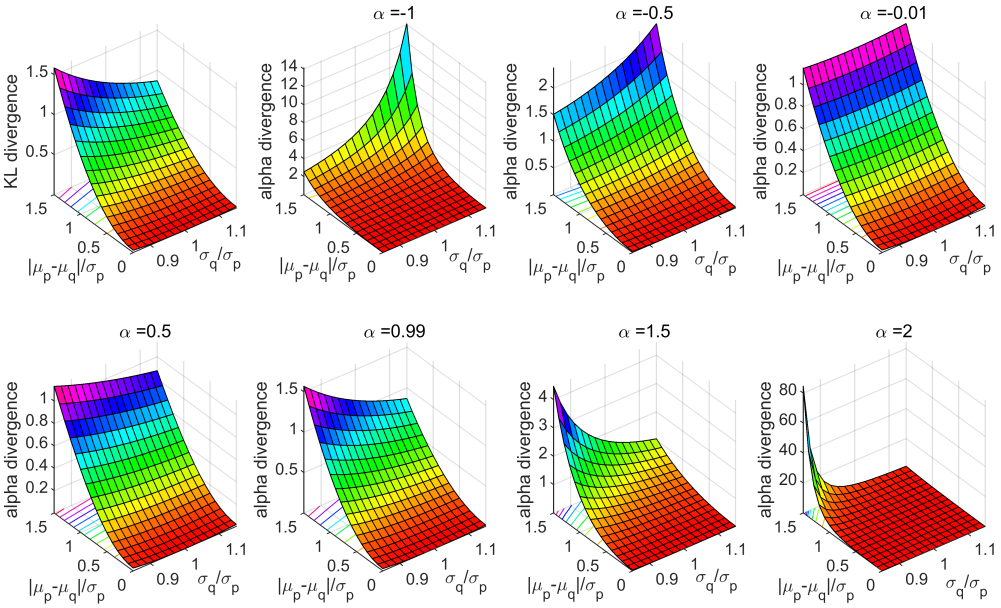
For two normalized densities, the alpha divergence [7, 27] is formulated as:

$$D_{AF}(p||q) = \frac{1}{\alpha(\alpha-1)}\left(\int p^{\alpha}(x)q^{1-\alpha}(x)dx - 1\right) \tag{6.12}$$

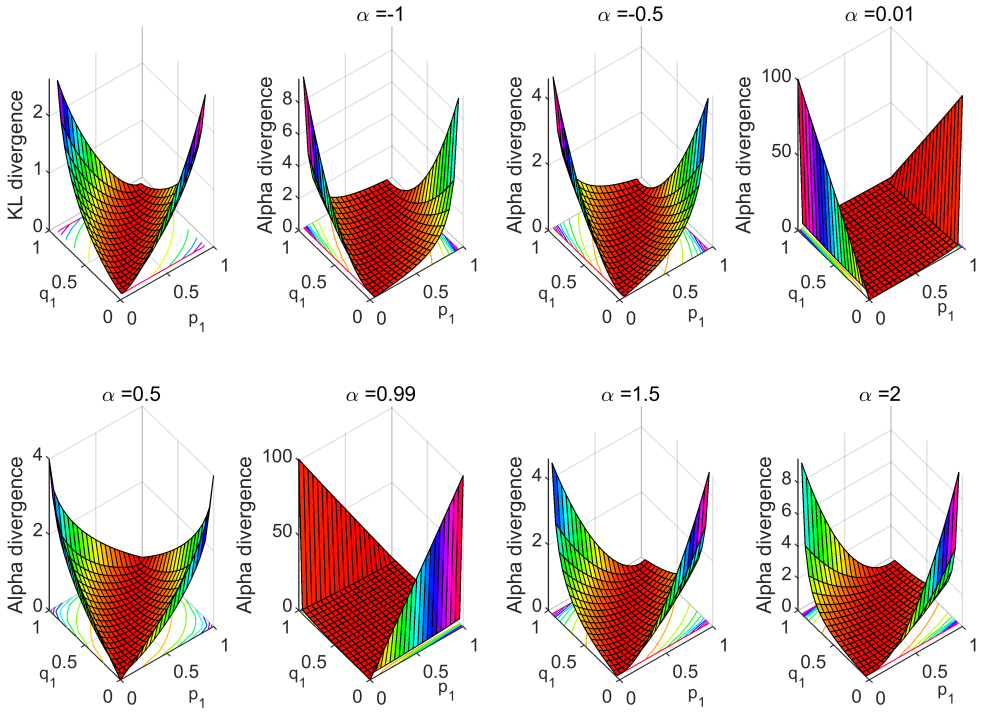
where $\alpha \neq 0$ and 1. This measure is also non-negative and asymmetric and achieves zero if and only if $p = q$ [7].

It is interesting that this alpha divergence covers some widely-mentioned divergences, as shown in Figure 6.5. For $\alpha = 2, 1/2$ and -1, we obtain Pearson Chi-square divergence, Hellinger distance and inverse Pearson or Neyman Chi-square divergence, i.e.,

$$\begin{aligned} D_{PC}(p||q) &= D_{AF}^{(2)}(p||q) = \tfrac{1}{2}\int \tfrac{(p(x)-q(x))^2}{q(x)}dx \\ D_{HR}(p||q) &= \tfrac{1}{2}D_{AF}^{(1/2)}(p||q) = \int (p(x)-q(x))^2 dx \\ D_{RP}(p||q) &= D_{AF}^{(-1)}(p||q) = \tfrac{1}{2}\int \tfrac{(p(x)-q(x))^2}{p(x)}dx. \end{aligned} \tag{6.13}$$



(a) Continuous case



(b) Discrete case

Figure 6.6: The KL divergence and alpha divergences with different alpha values.

For two singular values $\alpha = 1$ and 0, this alpha divergence would be defined as limiting cases for $\alpha \to 1$ and $\alpha \to 0$, respectively. When $\alpha \to 1$, such an alpha divergence would approximate the KL divergence. Additionally, if $\alpha \to 0$, we would derive the reverse KL divergence:

$$D_{KL}(q||p) = \lim_{\alpha \to 0} D_{AF}(p||q) = \int q(x) \log \frac{q(x)}{p(x)} dx. \tag{6.14}$$

Via embedding KL and inverse KL divergences, an extended alpha divergence is defined [27], i.e.,

$$D_{AD}(p||q) = \begin{cases} \frac{1}{\alpha(\alpha-1)} \left(\int p^{\alpha}(x) q^{1-\alpha}(x) dx - 1 \right), \alpha \neq 0, 1 \\ \int p(x) \log \frac{p(x)}{q(x)} dx, \alpha = 1 \\ \int q(x) \log \frac{q(x)}{p(x)} dx, \alpha = 0 \end{cases} \tag{6.15}$$

which has the following three basic properties:

(a) **Convexity**: this $D_{AD}(p||q)$ is convex with respect to p and q;

(b) **Strict positivity**: $D_{AD}(p||q) \geq 0$ and $D_{AD}(p||q) = 0$ if and only if $p = q$;

(c) **Continuity**: the extended alpha divergence (6.15) is continuous for the real variable α in the whole range including two singularities $\alpha = 0, 1$.

For a single continuous action, when $p(x) \sim N(\mu_p, \sigma_p^2)$ and $q(x) \sim N(\mu_q, \sigma_q^2)$, we have a closed representation:

$$D_{AF}(p||q) = \frac{1}{\alpha(\alpha-1)} \left(\frac{\sigma_p^{1-\alpha} \sigma_q^{\alpha}}{\sqrt{\alpha \sigma_q^2 + (1-\alpha)\sigma_p^2}} e^{\frac{1}{2} \frac{\alpha(\alpha-1)(\mu_p - \mu_q)^2}{\alpha \sigma_q^2 + (1-\alpha)\sigma_p^2}} - 1 \right). \tag{6.16}$$

Here, we introduce two new notions:

$$\begin{aligned} \rho_{\sigma} &= \sigma_q / \sigma_p \\ \rho_{\mu} &= |\mu_p - \mu_q| / \sigma_p \end{aligned} \tag{6.17}$$

to simplify the above two divergences as

$$D_{KL} = \log \rho_{\sigma} + \frac{1}{2} \rho_{\sigma}^{-2} + \frac{1}{2} \rho_{\mu}^2 - \frac{1}{2} \tag{6.18}$$

and

$$D_{AF} = \frac{1}{\alpha(\alpha-1)} \left(\frac{\rho_{\sigma}^{\alpha}}{\sqrt{\alpha \rho_{\sigma}^2 + (1-\alpha)}} e^{\frac{1}{2} \frac{\alpha(\alpha-1)}{\alpha \rho_{\sigma}^2 + (1-\alpha)} \rho_{\mu}^2} - 1 \right). \tag{6.19}$$

It could be observed that two divergences are associated with ρ_{σ} and ρ_{μ}. In Figure 6.6, we demonstrate KL divergence and alpha divergence with different alpha values.

In particular, when $\alpha < 0$, their surfaces are different from those with $\alpha > 0$ and KL divergence, which implies that we could achieve better performance from these special settings.

To guarantee $\alpha\rho_\sigma^2(1-\alpha) > 0$ in alpha divergence, we analyze three possible cases:

(i) $\rho_\sigma = 1$: we could use any real number, excluding 0 and 1.

(ii) $\rho_\sigma < 1$: we have

$$\alpha < \frac{1}{1-\rho_\sigma^2} > 1. \tag{6.20}$$

(iii) $\rho_\sigma > 1$: we derive

$$\alpha > -\frac{1}{\rho_\sigma^2 - 1} < 0. \tag{6.21}$$

According to the analysis of these three cases, we could achieve the most conservative interval for α in $[0,1]$, as shown in Figure 6.7, in which we also demonstrate the upper and bounds as a function of ρ_σ. For a real-world application, we will estimate a proper alpha interval via minimal and maximal ρ_σ during the training procedure.

Further, considering the widely existing multi-action situation, we extend alpha divergence to a more suitable form. It is usually assumed that the n continuous actions satisfy a n-variant normal distribution with a mean vector $\mu = [\mu_1, \mu_2, ..., \mu_n]^T$ and a diagonal covariance matrix (i.e., $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_n^2)$. In this setting, the alpha divergences have the following special forms:

$$D_{AD}(p||q) = \frac{1}{\alpha(\alpha-1)}\left(\prod_{i=1}^{n}\frac{\sigma_{p_i}^{1-\alpha}\sigma_{q_i}^{\alpha}}{\sqrt{\alpha\sigma_{q_i}^2+(1-\alpha)\sigma_{p_i}^2}}e^{\frac{1}{2}\frac{\alpha(\alpha-1)\left(\mu_{p_i}-\mu_{q_i}\right)^2}{\alpha\sigma_{q_i}^2+(1-\alpha)\sigma_{p_i}^2}} - 1\right) \tag{6.22}$$

Also, we provide the special forms of KL divergences in the same setting:

$$D_{KL}(p||q) = \sum_{i=1}^{m}\frac{1}{2}\left(\log\frac{\sigma_{q_i}^2}{\sigma_{p_i}^2} + \frac{\left(\mu_{p_i}-\mu_{q_i}\right)^2}{\sigma_{q_i}^2} + \frac{\sigma_{p_i}^2}{\sigma_{q_i}^2} - 1\right) \tag{6.23}$$

It is observed that the KL divergence is additive, while the alpha divergence is multiplicative, which shows the obvious difference between such two divergences.

6.3.3 An Improved PPO

In this sub-section, we will summarize our novel proximal policy optimization algorithm based on alpha divergence or alphaPPO. Now our optimization becomes a minimize

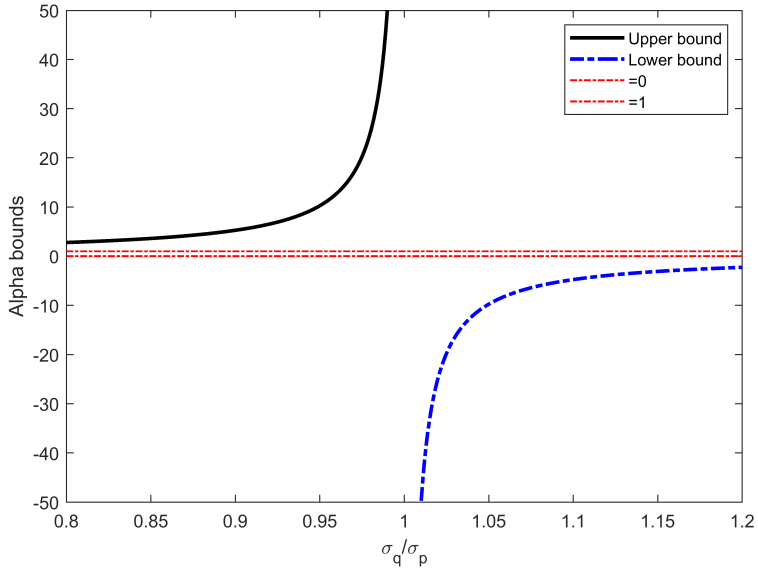


Figure 6.7: The alpha bounds for different ratios (σ_q/σ_p).

problem:

$$\min \ -(1-\beta)\hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\bar{\theta}}(a_t|s_t)}\hat{A}_t\right] + \beta\hat{\mathbb{E}}_t[D_{AF}(\pi_{\bar{\theta}}(a_t|s_t)||\pi_\theta(a_t|s_t))]. \tag{6.24}$$

We summarize our alphaPPO in **Algorithm 5**.

Algorithm 5 alphaPPO: PPO with alpha divergence

INITIALIZE: the **Actor-Critic** network architecture
the balanced factor β
the alpha values α
TRAIN:
for Iteration=1,2,... **do**
Collect a trajectory of $\{s_t, a_t, r_t\}$ via interacting with environment
using the policy π_θ
Calculate the advantages A_t according to (6.24)
Optimize the actor-critic network
Update the value function for critic network
Update the policy θ for actor network
end for

6.4 Experiments

In this section, the experiments will be conducted to demonstrate the effectiveness and efficiency of the proposed alphaPPO algorithm by comparing it with two other PPO-type

reinforcement learning methods (clipping and combined PPOs).

6.4.1 Algorithms, Settings and Environments

In this sub-section, we will first verify our alphaPPO, via comparing with clipping PPO and combined PPO, with two traditional control environments: Pendulum and Mountain Car, and four simulated robotics environments using the MuJoCo physics engine: Inverted Pendulum, Reacher, Swimmer and Hopper. The basic information for these environments is shown in Appendix.

The basic code for PPO based on Pytorch is downloaded from [1], which is used to implement combined PPO and our alphaPPO. For three compared algorithms, their actor and critic networks are built with a fully connected neural network with two hidden layers of size 128. The maximal number of episodes is set to 10000 for all the experiments, respectively. Next, we will tune the key parameters elaborately at first, and then detect the best performance for a fair comparison.

Table 6.1: The rewards of different ε parameters in clipping PPO for six environments

ε	Pendulum	Mountain Car	Inverted Pendulum	Reacher	Swimmer	Hopper
0.025		42.81				111.17
0.05	-307.00	**65.68**	827.40	-88.56	32.76	**165.01**
0.1	**-195.77**	44.17	911.78	-88.53	6.16	10.08
0.2	-279.62	56.88	947.03	-88.22	**34.44**	9.80
0.3	-210.39	37.05	**1000.0**	-87.02	7.49	10.06
0.4	-379.10	47.68	59.44	**-84.42**	30.00	9.67
0.5				-89.36		

6.4.2 Tuning Key Parameters for Compared Algorithms

We tuning clipping factor ε for clipping PPO. In clipping PPO, a default $\varepsilon = 0.2$ is widely recommended, for example in [34]. In order to validate whether this setting is optimal, we set ε with five different values: 0.05, 0.1, 0.2, 0.3 and 0.4, and the average rewards of last 100 episodes in each environment are shown in Table 6.1. If the optimal value is obtained at the edge, we would extend the range of ε slightly. In Table 6.1, the best returns are highlighted by **bold font** where the return is calculated over the last 100

[1] https://github.com/boyu-ai/Hands-on-RL

episodes. At the same time, we use Swimmer environment in MuJoCo as an example to show the tuning procedure in Figure 6.8.

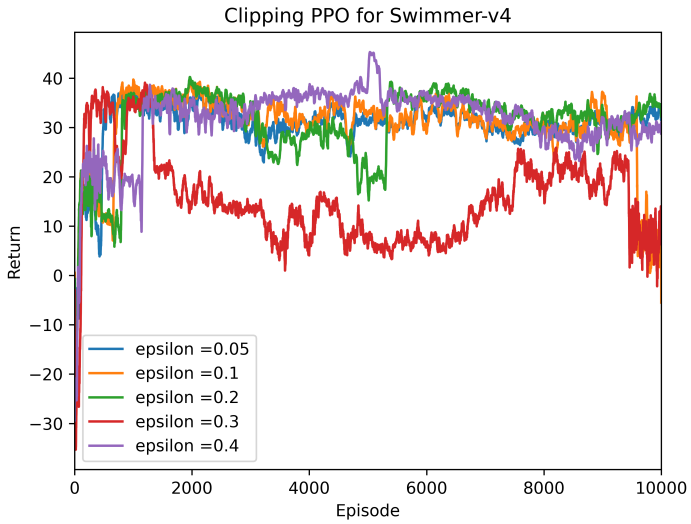


Figure 6.8: Tuning the ε value in clipped PPO for Swimmer

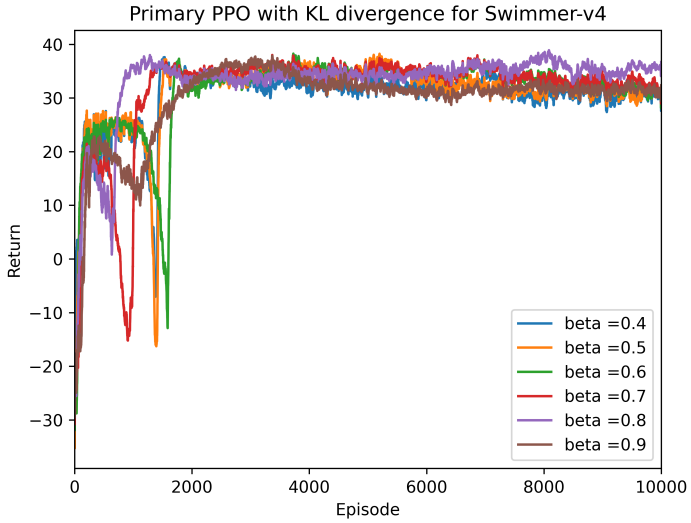


Figure 6.9: Tuning the balanced β value in combined PPO for Swimmer

The balanced factor β is tuned for combined PPO. For combined PPO, the balanced factor β is varied from 0.4 to 0.9 with a step of 0.1 in our experiments. The tuning

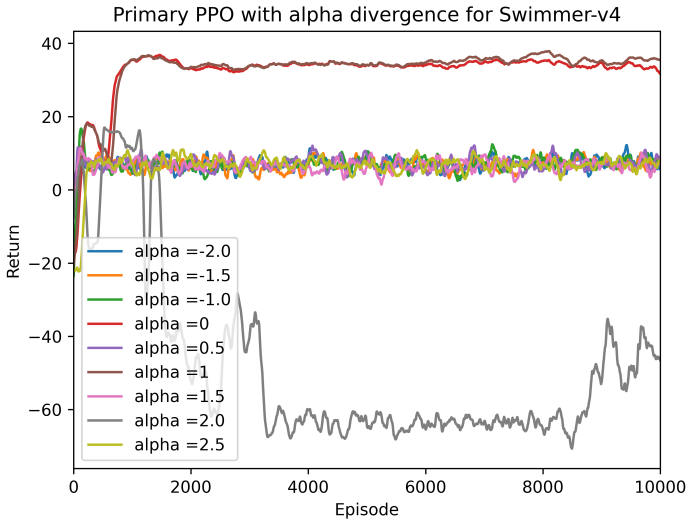


Figure 6.10: Tuning the α value in combined PPO for Swimmer

procedure is executed on Swimmer, as shown in Figure 6.9. The best tuning results of all six environments are highlighted in Table 6.2.

Table 6.2: The rewards of different β parameters for six environments in combined PPO

β	Pendulum	Mountain Car	Inverted Pendulum	Reacher	Swimmer	Hopper
0.4	-292.61	-6.41	975.78	-87.56	30.42	274.78
0.5	-269.92	89.74	**1000.0**	**-14.47**	31.13	284.27
0.6	-262.64	92.87	**1000.0**	-16.89	31.27	185.31
0.7	-226.38	36.37	954.65	-43.37	31.46	271.94
0.8	**-226.30**	87.84	**1000.0**	-87.18	**35.61**	278.21
0.9	-357.06	**93.99**	985.44	-16.36	30.95	**286.50**

Finally, the α for in our alphaPPO are determined. There are two key parameters: the α in alpha divergence and balanced factor β in combined PPO for our alphaPPO. Here, we apply a lazy strategy to search for an optimal α given an optimal β value from Table 6.2. Besides five specific alpha values in Figure 6.6, totally, we choose $\alpha = \{-2.0, -1.5, -1.0, 0$ $1.0, 1.5, 2.0, 2.5\}$ for each environment. Then, the tuning procedure of α in Swimmer environment is shown in Figure 6.10 as an example. Finally, we obtain the rewards of each α value in six environments and highlight the optimal α value as shown in Table

6.3. The optimal parameters for three methods and six environments are listed in Table 6.4. It is also observed that two environments (Inverted Pendulum and Swimmer) have a better performance with KL divergence ($\alpha = 1$).

Table 6.3: The rewards of different α parameters for six environments in alphaPPO

α	Pendulum	Mountain Car	Inverted Pendulum	Reacher	Swimmer	Hopper
-2.0	-1190.97	-75.86	984.32	-87.14	5.65	9.03
-1.5	-1183.64	34.66	947.30	-86.31	7.60	10.45
-1.0	-1281.46	27.64	992.94	-88.72	7.91	9.67
0	-173.56	51.96	993.68	**-13.78**	32.46	**452.06**
0.5	**-299.80**	**96.02**	963.92	-86.43	9.39	278.40
1.0	-226.38	92.87	**1000.00**	-14.47	**35.61**	185.31
1.5	-257.77	91.85	993.80	-88.26	6.89	297.47
2.0	-295.83	91.93	996.32	-85.91	-45.73	284.79
2.5	-257.22	88.22	901.87	-87.07	6.63	10.16

Table 6.4: The optimal parameters for three methods and six environments

Environment	ε in Clipping PPO	β in Combined PPO	α in alphaPPO
Pendulum	0.1	0.8	0.5
Mountain Car	0.05	0.9	0.5
Inverted Pendulum	0.3	0.5	1.0
Reacher	0.4	0.5	0
Swimmer	0.2	0.8	1.0
Hopper	0.05	0.9	0

6.4.3 Comparison of Three Algorithms

According to the above parameter tuning procedures, we determine the optimal key parameters for three compared methods (clipping PPO, combined PPO, and alphaPPO) in six environments, as shown in Table 6.4. Based on these optimal settings, we compare the three algorithms for each environment and the experiments of each algorithm run with four random seeds as shown in Figure 6.11. Table 6.5 also shows the average rewards of the last 100 episodes in each environment with one fixed random seed.

Table 6.5: The reward of three algorithms on six environments

Environment	Clipping PPO	Combined PPO	AlphaPPO
Pendulum	-195.77	-226.30	**-173.56**
Mountain Car	65.68	93.99	**96.02**
Inverted Pendulum	**1000.0**	**1000.0**	996.32
Reacher	-84.42	-14.47	**-13.78**
Swimmer	34.44	32.38	**34.88**
Hopper	165.01	284.2	**452.06**

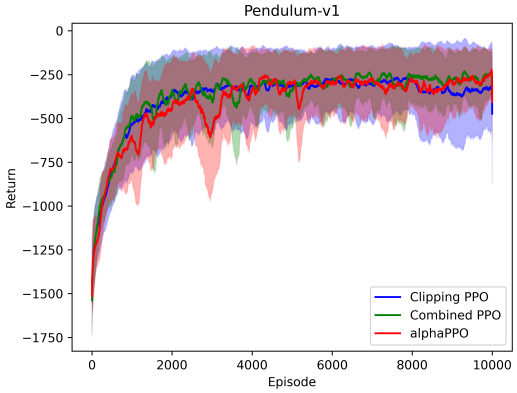
Table 6.6: Relative return based sample efficiency from three methods on six environments

Environment	Clipping PPO	Combined PPO	alphaPPO
Pendulum	0.7894	0.5000	**1.0000**
Mountain Car	0.5000	0.9665	**1.0000**
Inverted Pendulum	**1.0000**	**1.0000**	0.5000
Reacher	0.5000	0.9951	**1.0000**
Swimmer	0.9256	0.5000	**1.0000**
Hopper	0.5000	0.7076	**1.0000**
Mean	0.7025	0.7782	**0.9167**

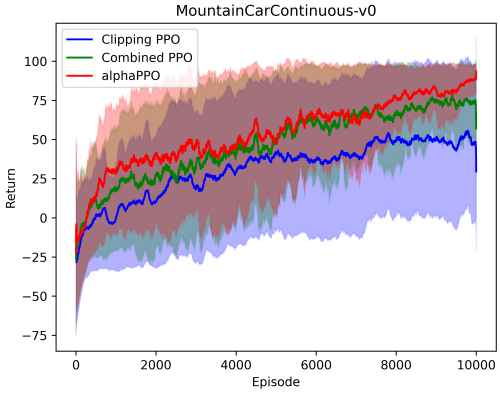
Our alphaPPO obviously outperforms two compared methods in the leftover subplots (a), (b), (e), and (f) of Figure 6.11. In Figure 6.11(d), three curves overlap to some extent, however, the reward value in Table 6.5 validates that alphaPPO is slightly superior to the other methods. According to Table 6.5 on Inverted Pendulum, it is illustrated that ouralphaPPO performs slightly worse than the other two methods on, but the difference is very small. We infer this tiny deterioration possibly comes from randomness. On the other hand, in Figure 6.11 (c), our alphaPPO becomes very stable after 4000 epochs, but clipping PPO oscillates and the combined PPO has a large deviation. This shows that our alphaPPO could still work well on this environment.

6.4.4 Performance Evaluation of Sample Efficiency

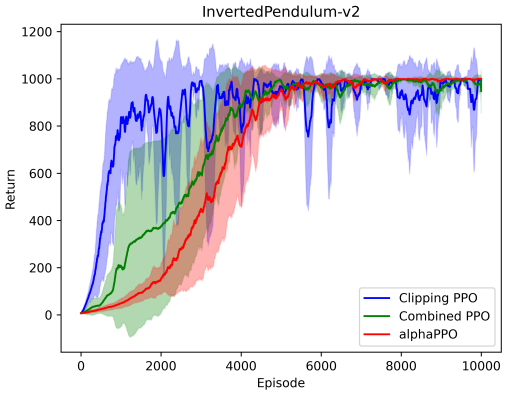
The AUC defined in chapter 4 is not good enough to compare two methods, since intuitively and visually the algorithm B is better than the algorithm A. In this case, we



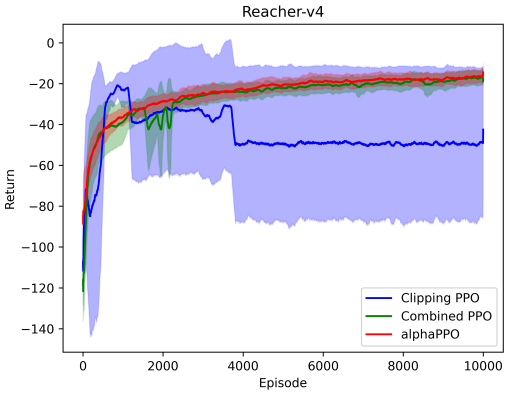
(a) Pendulum



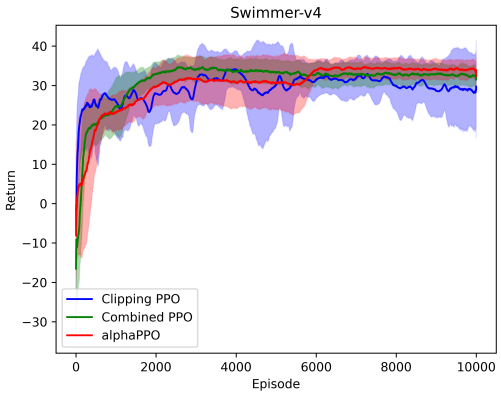
(b) Mountain Car



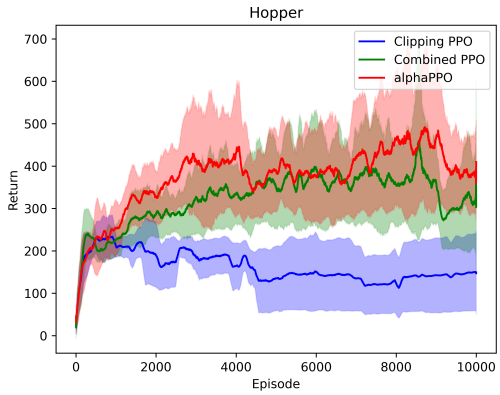
(c) Inverted Pendulum



(d) Reacher



(e) Swimmer



(f) Hopper

Figure 6.11: Comparison of PPO-type algorithms

define a relative return-based metric for sample efficiency:

$$SE_{RR} = \frac{1}{2}\left(\frac{R - R_{min}}{R_{max} - R_{min}} + 1\right) \in [0.5, 1.0] \tag{6.25}$$

where R indicates the average return obtained from the above sub-section, and R_{max} and R_{min} are the maximum and minimum of returns among several compared methods. Such a metric implies sample efficiency for a fixed number of episodes or a proper time step interval.

In Table 6.6, the sample efficiency based on relative return is estimated, which shows that our alphaTRPO is the most efficient in all environments but Inverted Pendulum. For the last row of Table 6.6, overall, the sample efficiency of our alphaTRPO is the highest among three compared methods.

In short, the aforementioned experiments validated the effectiveness and efficiency of our proposed alphaPPO algorithm.

6.5 Summary

In this chapter, we proposed an enhanced proximal policy optimization algorithm, which stems from the primary proximal policy optimization and substitutes alpha divergence for KL divergence. Experimentally, the effectiveness and the efficiency of our proposed method are validated using six benchmark environments and two compared algorithms.

CHAPTER 7

DIVERSITY-DRIVEN MODEL ENSEMBLE ADAPTIVE TRUST REGION POLICY OPTIMIZATION

In this chapter, we will propose a new model ensemble trust region policy optimization, via a novel deep neural network architecture for environment dynamics models and an adaptive Renyi alpha divergence for trust region policy optimization.

7.1 Introduction

In Chapter 3, we have reviewed some related work in model-ensemble reinforcement learning (MERL), from which we observed that:

(1) Shallow and wide neural networks are widely used to build ensemble-based dynamics models, in which the deep neural networks mean that ReLU is used. However, these true deep neural networks [43] have not been applied yet.

(2) So far, there are two implicit ways to reflect the model diversity: different initialization and different sample subsets. The explicit diversity is not investigated.

(3) TRPO is a popular policy learning technique, the improvement of which will enhance MERL approaches further.

These three observations inspire us to build a novel MERL approach in this thesis. We will propose a **D**iversity-driven model ensemble **A**daptive trust region policy optimization (DA-TRPO). Specifically, we firstly design a new network architecture based on U-net [102], residual network [50], and attention mechanism [131] to achieve better

representation ability than simple MLPs but with fewer neurons (or weights). It is widely recognized that diversity is an essential index for ensemble learning [77, 99]. Different from the naive methods used in previous works, like different initialization and different sample subsets, we develop a new diversity regularization technique to pursue the model diversity explicitly based on the Hilbert-Schmidt independence criterion which is a kernel-based index to measure the dependence between two sets of random variables [44]. Further, the success of model-based RL also depends on policy learning. TRPO [106] is one of the representative techniques in deep RL, whose primary form constrains its trust region of two consecutive policies via the parameter-free Kullback-Leibler (KL) divergence, and performance is greatly sensitive to its KL threshold. Instead, in this study, we apply a parametric Renyi alpha divergence [100, 130] and further adjust the alpha value adaptively during training iterations, which would benefit adaptive trust region detection and alpha value tuning. Finally, experiments on six benchmark environments show that our proposed method can achieve the best performance, compared with three existing RL methods. In summary, we propose three contributions in this chapter:

- we build a novel model ensemble RL technique containing a Hilbert-Schmidt independence criterion-based regularization term to pursue an explicit model diversity and well-designed base models to learn fewer network weights;
- our proposed adaptive Renyi alpha divergence detects the trust region adaptively between two adjacent policies for TRPO in the policy learning step.
- the effectiveness and efficiency of our proposed ensemble method are validated by comparing it with three State-of-the-art reinforcement learning techniques conducted on six benchmark environments.

The remainder of this chapter is organized as follows. We define our deep neural network architecture and its diversity-driven objective function, in Section 7.2 and 7.3, respectively. Then an adaptive trust region policy optimization is proposed in Section 7.4. The extensive experiments are provided and analyzed in Section 7.5. Finally, we end up this chapter with a concise summary.

7.2 Residual Attention U-nets for Dynamics Models

The existing dynamics models are constructed via feed-forward neural networks and their probabilistic forms, as shown in Figure 7.1 from ME-TRPO [63]. With four hidden

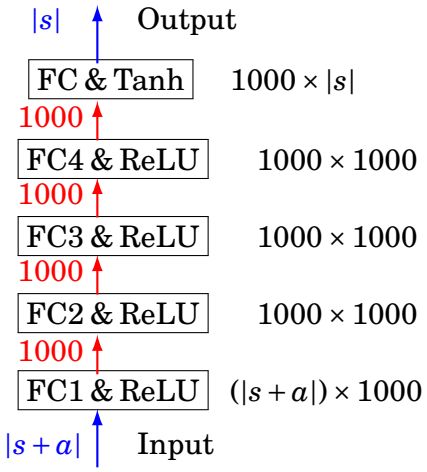


Figure 7.1: A feed-forward neural network with four hidden layers in ME-TRPO, where the black number indicates the size of weights for each layer and the red number is the output feature dimension of each layer. The + is the concatenation operation of two vectors and $|\cdot|$ indicates the length of the vector. For any benchmark environment, there are more than 1M and 3M weights for two and four hidden layers.

layers of 1000 neurons, for tradition control and MuJoCo environments, there are more than 3M weights to be optimized, which would result in a high computational complexity. To avoid the over-fitting problem, many interactions with the environment are still needed, which violates the primary aim of MBRL. To this end, we construct a novel network architecture in this subsection. At first, we introduce three modules: U-net, residual networks, and attention mechanism.

7.2.1 U-net

An autoencoder is a specific type of neural network, which cascades an encoder with a decoder. The former is designed to encode the input into a compressed and meaningful representation, and the latter decodes it back such that the reconstructed input is as similar as possible to the original [9, 129]. This network is originally implemented by fully connected (FC) layers and then induces a famous encoder-decoder architecture in deep learning [43].

For the image segmentation task, after the FC layers are replaced by convolutional layers to construct a fully convolutional network (FCN), the U-net [102] particularly adds some skipping connections to concatenate low levels features to high-level features. This kind of connection not only facilitates back-propagation during training but also

compensates low-level finer details for high-level semantic information. Nowadays, U-net and its variants have become effective tools for image segmentation and related tasks [111].

In this study, we will build our dynamics model with U-net with all FC layers, as our backbone network.

7.2.2 Residual Network

As the number of layers increases, when training a neural network or deep learning network using a back-propagation algorithm, it can happen that the gradients vanish and explode. To address this issue, a residual network is proposed in [50], which is composed of several residual blocks. Each residual block consists of 3-4 convolutional layers, in which the input is added to the output of the last layer via a shortcut addition connection. Similarly, as concatenation in U-net, this addition connection also propagates the low-level information to the high levels. It has been shown that this residual trick can effectively speed up the training procedure and further enhance the classification performance, especially for image recognition [108].

In our study, we construct each residual module with FC layers and ReLU activation functions, as shown in Figure 7.2. The purpose of the first FC layer would enlarge the length of input doubly, and the next three layers remain unchanged in feature dimension. We build a shortcut connection between the output of the first layer and that of the last layer since the addition operation needs the same length.

The skipping concatenation connection in U-net is from the output of the fourth layer, and the low-level attention comes from the addition operation, which implies that the concatenation and attention exploit different low-level information, as shown in Figure 7.2.

7.2.3 Attention Mechanism

The attention mechanism is mostly popular in natural language processing (NLP) [131], which pays more attention to the subset of its input. Moreover, it has been employed in many applications [18, 88], for example, computer vision. The attention mechanism detects which parts of the network require more attention in the neural network. The attention mechanism also reduces the computational cost of encoding the information in each image into a vector of fixed dimensions. The main strength of the attention

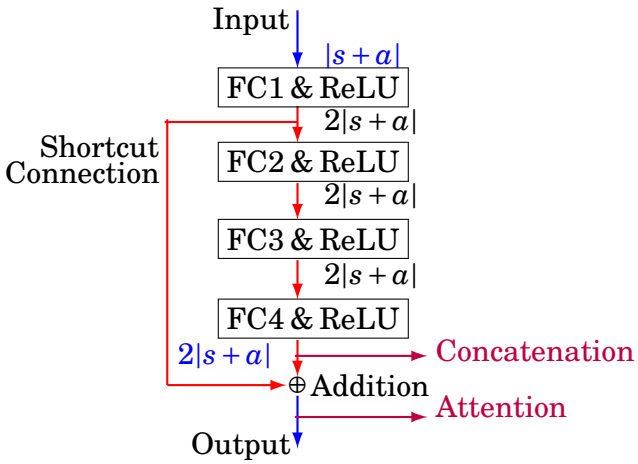


Figure 7.2: Residual module used in this study, where the length of feature vector indicates the **Res module 1** in Figure 7.5.

mechanism is that although it is simple, it can be applied to any input size, and enhances the quality of features that boost the results further.

In [54], an attention mechanism is designed according to low-level and high-level information for image segmentation. In this study, we simplify their implementation via substituting FC layers for convolutional layers, as shown in Figure 7.3. Here, the first FC layer is to transform the low-level features into the same length as the low-level features, since both addition and multiplication operations need the same length vectors.

Sequentially, we combine the attention mechanism and residual block to build the residual attention (ResAtt) module as shown in Figure 7.4, which aims to shorten the length of feature vectors by half overall. The first two FC layers would reduce the length of their input features since the attention and concatenation operations would produce a longer feature vector.

7.2.4 Residual Attention U-net for Ensemble Dynamics Model

Finally, we integrate the aforementioned U-net architecture, residual module, and residual attention module to construct a residual attention U-net, or simply RauNet, as shown in Figure 7.5. There are four components: encoder, bridge, decoder, and output. The encoder includes two residual modules, and the decoder covers two residual attention modules. The bridge component connects the encoder with the decoder. It is noted that this component excludes concatenation and attention connections. The output module

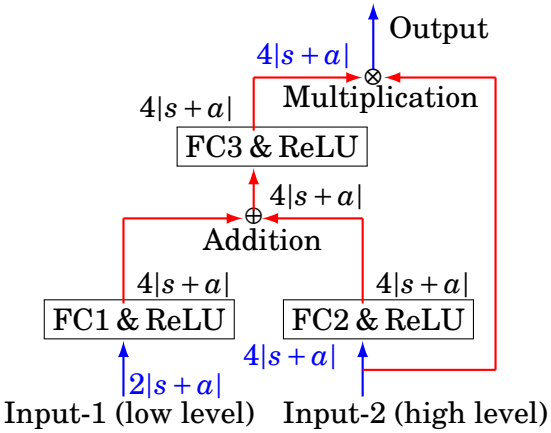


Figure 7.3: Attention mechanism module, where the length of the feature vector is from **Attention module** in Figure 7.4.

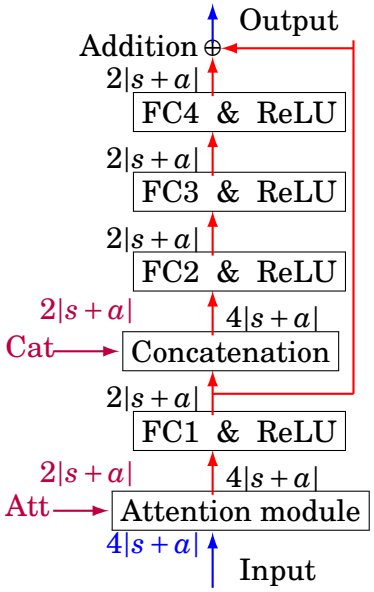


Figure 7.4: Residual attention architecture, where the length of feature vector indicates in **ResAtt module 1** in Figure 7.5.

consists of two FC layers, as shown in Figure 7.6. In Figure 7.5, we indicate the output vector length of each module, where $s+a$ indicates the concatenation operation of two vectors and $|\cdot|$ is the length of the vector.

When the deep neural network in Figure 7.1 is built using m hidden layers and h

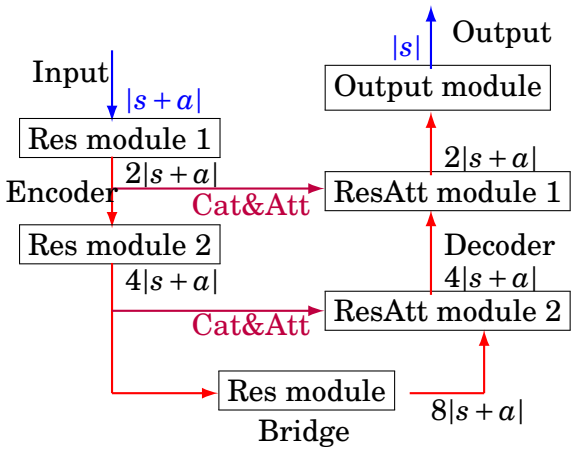


Figure 7.5: Residual attention U-net (RauNet) proposed in this study.

Table 7.1: Estimation of weight size for our RauNet

Module	Weights
Res Module 1	$14(\|s\|+\|a\|)^2$
Res Module 2	$56(\|s\|+\|a\|)^2$
Res Module	$224(\|s\|+\|a\|)^2$
ResAtt Module 2	$288(\|s\|+\|a\|)^2$
ResAtt Module 1	$72(\|s\|+\|a\|)^2$
Output Module	$2(\|s\|+\|a\|)^2+\|s\|(\|s\|+\|a\|)$
Overall	$656(\|s\|+\|a\|)^2+\|s\|(\|s\|+\|a\|)$

neurons, except for bias terms, the number of weights is estimated as follows:

$$\text{DNN} = (m-1)h^2 + 2h|s| + h|a| \approx (m-1)h^2 \tag{7.1}$$

which shows that a four hidden layer network has more than 3M weights.

For our RauNet, the number of weights is estimated in Table 7.1, where the total number of weights is:

$$\text{RauNet} = 656(|s|+|a|)^2 + |s|(|s|+|a|) \approx 656(|s|+|a|)^2. \tag{7.2}$$

When $|a|+|s| = 40$, the weight size is close to 1M, which is true for many benchmark environments. Therefore, our RauNet is a lightweight deep network.

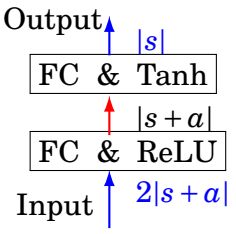


Figure 7.6: Output module used in Figure 7.5.

7.3 Diversity-driven Objective Function

The aim of MFRL is to learn an optimal policy π that maximizes the aforementioned $J(\pi)$, or the value function $V^{\pi}(s_t)$. However, in MBRL, we assume access to an analytic differentiable transition function (and reward function), i.e., an environment dynamics model [81, 92]. After collecting some interactions with the true environment, the experienced transitions are denoted as follows:

$$D = \{(s_t, a_t, s_{t+1})\} \tag{7.3}$$

which is usually stored in a replay buffer of a fixed length and thus is sequentially updated over time. We use this date (7.3) to learn a dynamics function f. When the ground-truth dynamics are deterministic, the learned dynamics function f predicts the next state from the current state and action. In stochastic settings, it is usual to represent the dynamics with a Gaussian distribution, i.e., $p(s_{t+1}|a_t, s_t) \sim \mathcal{N}(\mu(s_t, a_t), \Sigma(s_t, a_t))$, in particular, the covariance matrix Σ is simply diagonal [81, 92]. For ensemble models, a set of m dynamics functions $f = \{f_i | i = 1, ..., m\}$ need to be trained.

In ensemble learning [77, 99], diversity is one of the key factors that must be considered. However, as pointed out in the related work in Chapter 2, existing model ensemble RL methods only apply different initialization and different sample subsets, which generally are regarded as implicit diversity.

In policy ensemble learning, to find out a set of diverse policies, maximum mean discrepancy (MMD) and orthogonality are used as regularization terms in [74] and [148], respectively, which measure the difference between two policies.

In deep learning, the similarity of neural network representation is revisited in [61], listing several famous metrics, for example, Hilbert-Schmidt independence criterion (HSIC) and its normalized version (NHSIC) or centered kernel alignment (CKA), canonical correlation analysis (CCA) and its singular vector form (SVCCA) and so on.

In this research, we will explicitly enforce a model diversity for neural network parameters via NHSIC. The original HSIC is a kernel-based index to measure the dependence between two sets of random variables. When two kernels are universal, HSIC=0 indicates independence. Let two neural networks be parameterized using ϕ_i and ϕ_j, respectively. Since each different FC layer has its number of weights and bias terms, we arrange and reshape those parameters of each network into a long vector for convenience. Assume that each neural network corresponds to a kernel matrix K_i. Then, the empirical estimator between two networks is defined as:

$$\text{HSIC}(K_i, K_j) = \frac{1}{(n-1)^2} trace\left(K_i H K_j H\right) \tag{7.4}$$

where n is the total number of parameters of each network, "trace" is the trace operation, and H is the centering matrix.

Since HSIC is not invariant to isotropic scaling, its invariant is defined as

$$\text{NHSIC}(K_i, K_j) = \frac{\text{HSIC}(K_i, K_j)}{\sqrt{\text{HSIC}(K_i, K_i)\text{HSIC}(K_j, K_j)}} \tag{7.5}$$

which is referred to as normalized HSIC (i.e., NHSIC) [147] and centered kernel alignment (CKA) in [28, 30].

According to this NHSIC, we define a diversity index for m dynamics models as follows:

$$\mathscr{L}^{div} = \frac{1}{m(m-1)} \sum_{i,j=1, i\neq j}^{m} \text{NHSIC}(K_i, K_j). \tag{7.6}$$

Since we train each model separately, this representation for the i-th model could be simplified as

$$\mathscr{L}_i^{div} = \frac{1}{m-1} \sum_{j=1, j\neq i}^{m} \text{NHSIC}(K_i, K_j). \tag{7.7}$$

Finally, for the i-th dynamics model, its diversity-driven objective function becomes

$$\mathscr{L}_i = \mathscr{L}_i^{mse} + \mu \mathscr{L}_i^{div} \tag{7.8}$$

where μ is a positive constant for diversity-driven regularization term, and the mean squared error term $\mathscr{L}_i^{mse}$ is defined as:

$$\mathscr{L}_i^{mse} = \frac{1}{|D|} \left\| s_{t+1} - f_{\phi_i}(s_t, a_t) \right\|_2^2. \tag{7.9}$$

According to the replay buffer data (7.3). In this case, our novel objective function (7.8) considers the model diversity explicitly, which would benefit to the model bias and variance reduction.

7.4 Adaptive Trust Region Policy Optimization with Renyi Alpha Divergence

TRPO [106] is one of the representative actor-critic architecture RL methods [34], whose actor network is to learn a policy and the critic network is to fit the value function.

Assume that the policy π is parameterized by a parameter vector θ, which is indicated by π_θ. With the previous policy $\pi_{\theta_{old}}$, the agent interacts with the simulated environment to collect some fictitious trajectory data, and then the TRPO optimizes a constrained maximization problem:

$$\begin{aligned} &\max \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A(s_t, a_t) \right] \\ &\text{s.t.} \quad \hat{\mathbb{E}}_t[D_{KL}(\pi_\theta(a_t|s_t)||\pi_{\theta_{old}}(s_t|a_t))] \leq \delta \end{aligned} \tag{7.10}$$

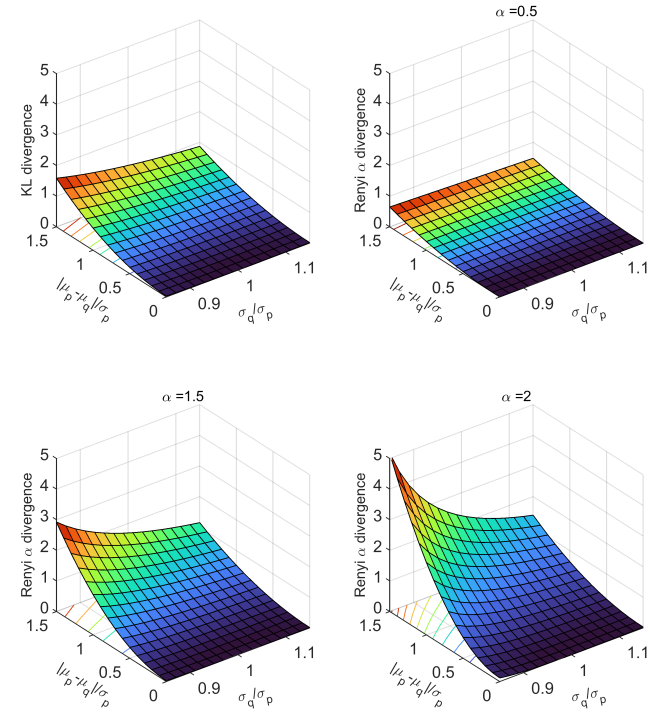
where θ and θ_{old} are two parameter vectors corresponding to the current and previous policies, respectively; the $D_{KL}(\cdot||\cdot)$ indicates the Kullback-Leibler (KL) divergence (or relative entropy) [29, 130]; δ is the threshold of KL divergence; and $A(a_t, s_t)$ the advantage function, please refer to their detailed definitions in Chapter 2.

In Chapter 2, the KL divergence and its special representation for two Gaussian distributions are summarized. Such a KL divergence is a parameter-free divergence, as a result, the performance of TRPO is sensitive to the threshold δ since the trust region is fixed during the whole TRPO learning procedure. In this study, we will design and implement an adaptive trust region.

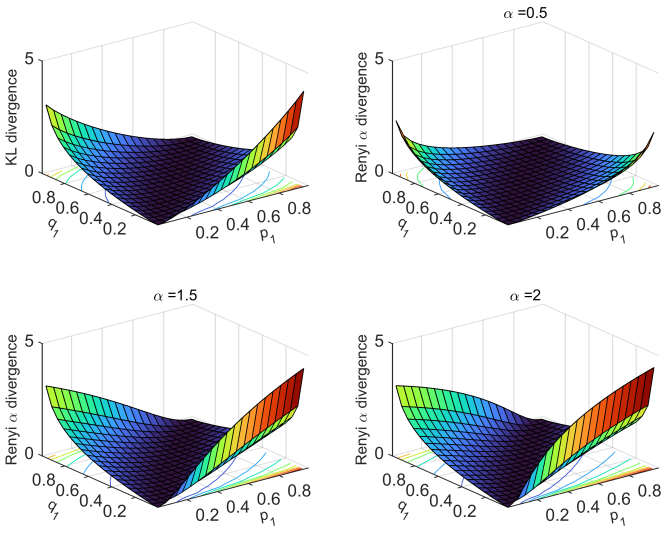
To characterize the difference between two probability distributions, some parametric divergences have been proposed [27]. In this study, we will consider Renyi alpha divergence, [49, 100, 130], which is defined as

$$D_{RA}^{(\alpha)}(p||q) = \frac{1}{\alpha - 1} \ln \int p(x)^\alpha q(x)^{1-\alpha} dx, \alpha > 0, \alpha \neq 1. \tag{7.11}$$

There are three special α values, which correspond to three existing divergences or distances: (a) Hellinger distance: $\alpha = 1/2$; (b) KL divergence: $\alpha \to 1$; and (c) Chi-square divergence: $\alpha = 2$.



(a) Continuous case



(b) Discrete case

Figure 7.7: KL divergence and Renyi alpha divergence for three special α settings.

Interestingly, the Renyi alpha divergence has the following useful properties:

(a) **Non-decreasing**: $D_{RA}^{(\alpha)}(p||q)$ is non-decreasing in α, often strictly so.

(b) **Continuity**: $D_{RA}^{(\alpha)}(p||q)$ is continuous on $\alpha \in [0,\infty]$.

(c) **Positivity**: $D_{RA}^{(\alpha)}(p||q) \geq 0$ for $\alpha \in [0,\infty]$, often strictly so.

(d) **Convexity**: $D_{RA}^{(\alpha)}(p||q)$ is jointly convex in (p,q) for $\alpha \in [0,1]$, convex in q for $\alpha \in [0,\infty]$, and jointly quasi-convex in (p,q) for $\alpha \in [0,\infty]$.

For two univariate Gaussian distributions, we have a closed form:

$$D_{RA}^{(\alpha)}(p||q) = \frac{1}{2}\frac{\alpha(\mu_p-\mu_q)^2}{\alpha\sigma_q^2+(1-\alpha)\sigma_p^2} + \frac{1}{\alpha-1}\ln\frac{\sigma_p^{1-\alpha}\sigma_q^{\alpha}}{\sqrt{\alpha\sigma_q^2+(1-\alpha)\sigma_p^2}}. \tag{7.12}$$

In Figure 7.7(a), we illustrate KL divergence and Reny alpha divergence for some special α settings for different Gaussian distributions. Further, we analyze two special cases: (a) $\sigma_p = \sigma_q$:

$$D_{RA}^{(\alpha)}(p||q) = \frac{1}{2}\left(\frac{\mu_p-\mu_q}{\sigma_p}\right)^2\alpha \tag{7.13}$$

which is linear with respect to α. (b) $\mu_p = \mu_q$:

$$D_{RA}^{(\alpha)}(p||q) = \frac{1}{\alpha-1}\ln\frac{\sigma_p^{1-\alpha}\sigma_q^{\alpha}}{\sqrt{\alpha\sigma_q^2+(1-\alpha)\sigma_p^2}} \tag{7.14}$$

which is non-linear with respect to α. In Figure 7.8, we show the Renyi alpha divergence as a function of α, for three different cases, which validates its non-decreasing property.

Considering two n-variate Gaussian distributions, we also have

$$\begin{aligned} D_{RA}^{(\alpha)}(p||q) &= \sum_{i=1}^{n}\left(\frac{1}{2}\frac{\alpha(\mu_{pi}-\mu_{qi})^2}{\alpha\sigma_{qi}^2+(1-\alpha)\sigma_{pi}^2} + \frac{1}{\alpha-1}\ln\frac{\sigma_{pi}^{1-\alpha}\sigma_{qi}^{\alpha}}{\sqrt{\alpha\sigma_{qi}^2+(1-\alpha)\sigma_{pi}^2}}\right) \\ &= \sum_{i=1}^{n} D_{RA}^{(\alpha)}(p_i||q_i). \end{aligned} \tag{7.15}$$

This relation implies that the Renyi alpha divergence is additive, just as the KL divergence.

For two discrete distribution laws: $p = [p_1,...,p_n]$ and $q = [q_1,...,q_n]$, their KL and Renyi alpha divergences are defined as:

$$\begin{aligned} D_{KL}(p||q) &= \sum_{i=1}^{n} p_i ln\left(\frac{p_i}{q_i}\right) \\ D_{RA}^{(\alpha)}(p||q) &= \frac{1}{\alpha-1} ln \sum_{i=1}^{n} p_i^{\alpha} q_i^{1-\alpha}. \end{aligned} \tag{7.16}$$

We illustrate the KL divergence with two discrete variates in Figure 7.7(b).

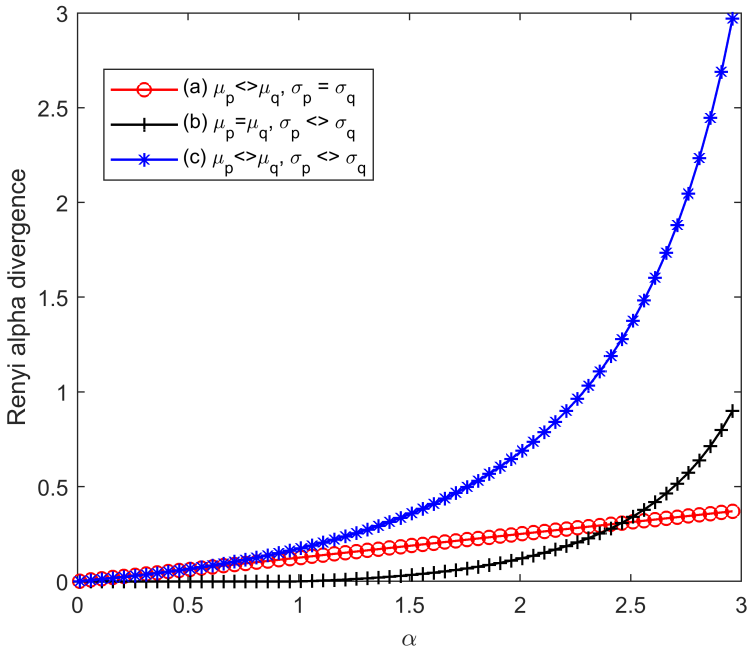


Figure 7.8: Nondecreasing property of Renyi alpha divergence with respect to α.

The Renyi alpha divergence can better measure the difference between two successive policies since the trust region of TRPO is related to the alpha value. However, its optimal alpha value needs to be searched for via grid scanning, which is time-consuming.

From the above nondecreasing property, we find out that, for two fixed distributions, as the alpha increases, the divergence becomes larger and approximates the given divergence threshold δ, which implies that the exploration ability is limited.

Therefore, according to this observation, we design a mechanism to adaptively adjust the alpha value for Renyi alpha divergence in each iteration.

Let the average return of the ith iteration be $\bar{R}_i$ $(i = 0, 1, 2, ...)$. We consider three cases:

(a) Increasing case: $\bar{R}_I > (1+\tau)\bar{R}_{i-1}$, where $\tau \in (0, 1]$ is a positive small constant, i.e., the average return is increasing with $\tau\bar{R}_{i-1}$ at least. In this case, the alpha value is unchanged, i.e., $\alpha_i = \alpha_{i-1}$.

(b) Fluctuating case: if $\bar{R}_I \in [(1-\tau)\bar{R}_{i-1}, (1+\tau)\bar{R}_{i-1}]$, we reduce the alpha value to enhance the exploration ability.

(c) Decreasing case: when $\bar{R}_i < [(1-\tau)\bar{R}_{i-1}$, we increase the alpha value to compress the exploration ability.

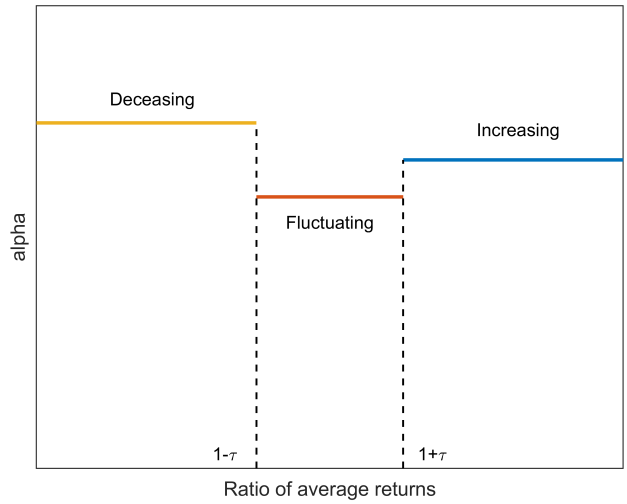


Figure 7.9: A diagram to show how to adjust the alpha value.

Finally, we summarize the above considerations as the following rules:

$$\alpha_i = \begin{cases} \alpha_{i-1} + \frac{\Delta\alpha}{\alpha_{i-1}+1}, & \text{if } \frac{\bar{R}_i}{\bar{R}_{i-1}} < 1-\tau \\ \alpha_{i-1} - \frac{\Delta\alpha}{\alpha_{i-1}+1}, & \text{if } 1-\tau \leq \frac{\bar{R}_i}{\bar{R}_{i-1}} \leq 1+\tau \\ \alpha_{i-1}, & \text{if } \frac{\bar{R}_i}{\bar{R}_{i-1}} > 1+\tau \end{cases} \tag{7.17}$$

where $\Delta\alpha$ is an increment for α. In our experiments, we set them as $\alpha_0 = 1.0$, $\Delta\alpha = 0.1$ and $\tau = 0.05$, and show an illustrative diagram in Figure 7.9. This process essentially adjusts the trust region adaptively.

In [106], the basic convergence of TRPO with KL divergence is analyzed in detail. Since the mathematical properties are almost the same between the KL divergence and the Renyi alpha divergence, our adaptive alpha TRPO also has a good convergence.

Finally, the performance of TRPO is validated using the m learned dynamics models. An accumulative reward is defined as:

$$\eta(\theta,\phi) = \hat{\mathbb{E}}_\tau[\sum_{t=0}^{H} r(s_t,a_t,s_{t+1})] \tag{7.18}$$

where $\hat{\tau} = \{a_0,s_0,a_1,s_1,...\}$, $s_0 \sim \rho_0(\cdot)$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} = f(s_t,a_t)$. Then, the ratio of models in which the policy improves is estimated as follows:

$$\frac{1}{m}\sum_{k=1}^{m} I(\eta(\theta_{new},\phi_k) > \eta(\theta_{old},\phi_k)) \tag{7.19}$$

Algorithm 6 Diversity-driven model ensemble algorithm with adaptive trust region policy optimization (DA-TRPO)

INITIALIZATION
  Initialize a policy π,
  m RauNet dynamics models $f_1, f_2, ..., f_m$,
  $\alpha = 1.0$,
  an empty dataset D.
PROCESS
  Repeat
    Collect samples from the real system f using π and add them to D.
    Compute the average return
    Adjust the alpha value
    Train all RauNet-based models using D
      via minimizing diversity-driven objective.
    Repeat
      Collect fictitious samples from $\{f_k | k = 1, ..., m\}$ using π.
      Update the policy using adaptive alpha TRPO on the fictitious samples.
      Estimate the performance for each model.
    Until the performances stop improving.
  Until the policy performs well in real environment f.

where $I(\cdot)$ is an indicator function, in which $I(true) = 1$.

The current iteration continues as long as this ratio exceeds a certain threshold. This process continues until the desired performance in the real environment is reached.

7.5 Diversity-driven Model Ensemble Adaptive Trust Region Policy Optimization

According to the above three sub-sections, we summarize our proposed novel diversity-driven model ensemble adaptive trust region policy optimization in **Algorithm 6**, which is referred to as DA-TRPO. This algorithm combines residual attention U-net, diversity-driven objective function, and adaptive Renyi alpha divergence.

From the aforementioned Algorithm 6, we could also derive two simplified versions as shown in Table 7.2: (a) RN-TRPO, which only embeds our proposed new deep neural network RauNet and (b) RN-aTRPO, which adds Renyi alpha divergence. In this next section, we will validate our proposed DA-TRPO.

Table 7.2: DA-TRPO and its two reduced forms

Algorithm	RauNet	Diversity	TRPO	
			Renyi alpha	Adaptive
RN-TRPO	✓			
RN-aTRPO	✓		✓	
DA-TRPO	✓	✓	✓	✓

Table 7.3: The basic information for six environments

Environment	State space	Action space	Weights in DNN	Weights in RauNet	Ratio
Cartpole	4	2	3.010M(4)	0.024M	0.0080
Pendulum	3	1	3.007M(4)	0.010M	0.0033
InvertedPendulum	4	1	3.009M(4)	0.016M	0.0053
Swimmer	8	2	3.018M(4)	0.066M	0.0219
Half-cheetah	17	6	3.040M(4)	0.347M	0.1142
Walker2d	17	6	3.040M(4)	0.347M	0.1142

7.6 Experiments

In this section, some detailed experiments on six environments are conducted to demonstrate the effectiveness of our proposed DA-TRPO, by comparing them with the original TRPO [106], single model-based method MB-MF [86], and model ensemble method ME-TRPO [63].

7.6.1 Environments and Compared Algorithms

To experimentally verify our proposed algorithms: DA-TRPO, and three existing algorithms: TRPO, MB-MF, and ME-TRPO, six benchmark environments will be investigated, which cover two traditional control environments: Cart Pole and Pendulum, and four simulated MuJoCo environments: Inverted Pendulum, Swimmer, Half Cheetah, and Walker2d, as shown in Appendix.

In Table 7.3, we list the number of weights from DNN with four hidden layers in ME-TRPO and RauNet in our proposed methods. It is found that the network of our models is much smaller than that of DNN, as shown in the last column: the ratio of RauNet weight size compared with a DNN with four hidden layers.

Table 7.4: The average return of all algorithms on six environments

Environment	TRPO	MB-MF	ME-TRPO	DA-TRPO
Cart Pole	-30.89(±70.65)	55.82(±19.86)	142.39(±76.32)	**186.23(±19.77)**
Pendulum	162.59(±4.93)	157.64(±13.45)	164.40(±4.02)	**167.18(±3.87)**
Inverted Pendulum	-104.02(±16.85)	-156.15(±25.00)	-26.69(±23.82)	**-12.31(±8.00)**
Swimmer	26.31(±7.49)	**30.14(±2.82)**	3.36(±19.34)	21.75(±7.17)
Half-cheetah	-352.01(±24.40)	-475.44(±40.70	119.27(±10.09)	**137.05(±42.51)**
Walker2d	-2771.11(±161.09)	-2705.52(±167.09)	-2712.07(±127.33)	**-318.53(±33.48)**

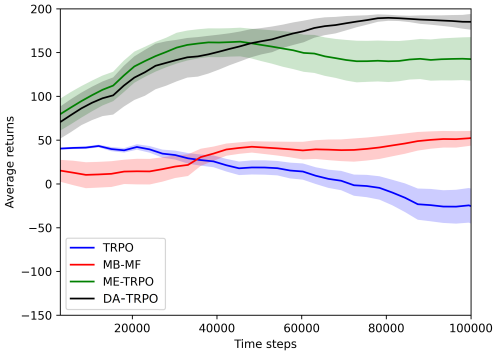
For the four compared algorithms (TRPO, MB-MF, ME-TRPO, and DA-TRPO), the first three existing algorithms TRPO, MB-MF, ME-TRPO are from [1] [133], and are executed with their default settings as much as possible. However, to fit our computational resource, the sampled trajectory data are divided into training and validating samples equally, the size of each buffer for training and validating dynamics model is 20K, and the batch size is 512. We also adjust the number of iterations to reach 100K time steps. For CartPole, Pendulum, and Inverted Pendulum, their learning rate is 0.01, otherwise 0.001. Except for our new deep network RauNet, our algorithms are set to have similar parameters as ME-TRPO. Specifically, in DA-TRPO, the α is initialized as $\alpha_0 = 1$ and is updated automatically at each iteration, and the default set of μ is 10 except Pendulum where the μ is 100. After the network parameters are normalized to follow the standard Gaussian distribution, the width of the RBF kernel is set to 1 for estimating kernel matrices in NHSIC.

7.6.2 Experimental Comparison and Analysis for Six Algorithms

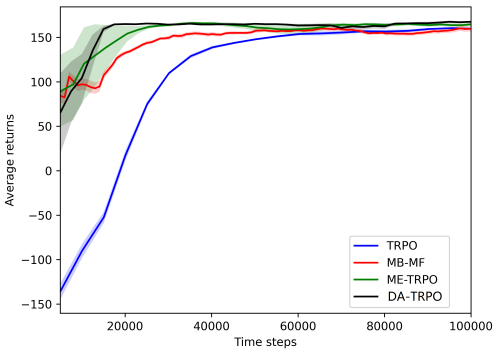
According to the aforementioned settings, we conduct the experiments on all environments with ten random seeds, as shown in Figure 7.10.

In Figure 7.10 (a) (c) (e) and (f), our DA-TRPO performs more effectively than other methods on four environments: Cart Pole, Inverted Pendulum, Half Cheetah, and Walker2d. In the environment Pendulum, our DA-TRPO and other methods perform similarly as shown in Figure 7.10(b). Although the TRPO and MB-MF methods are superior to our algorithms on the environment Swimmer in Figure 7.10(d), the proposed DA-TRPO method achieves better returns than the ME-TRPO, which is considered a valid improvement.

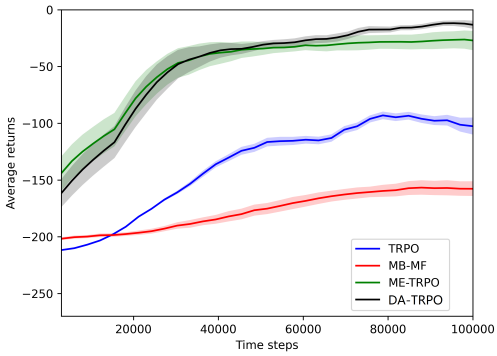
[1] https://github.com/WilsonWangTHU/mbbl-metrpo



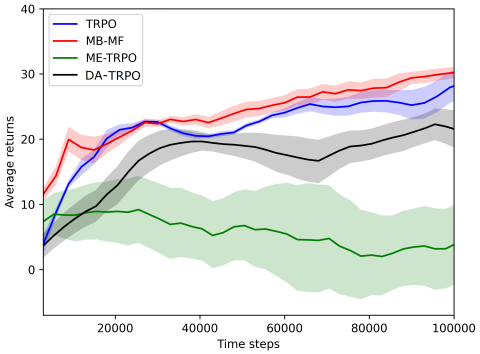
(a) Cart Pole



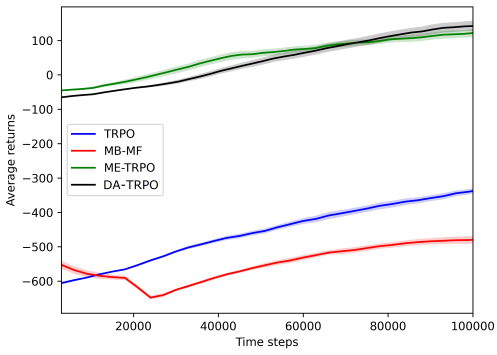
(b) Pendulum



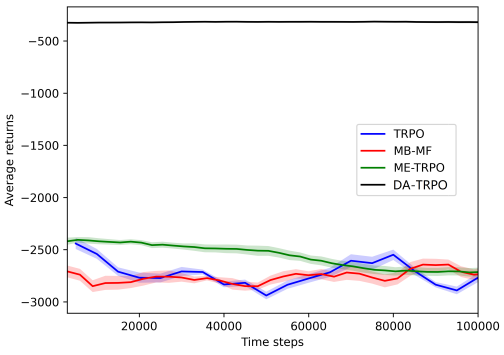
(c) Inverted Pendulum



(d) Swimmer



(e) Half-cheetah



(f) Walker2d

Figure 7.10: Comparison of four reinforcement learning algorithms on six environments

Further, we calculate the final returns of each algorithm in each environment, which represent the average and standard deviation of the returns of the last five iterations, as shown in Table 7.4. Our proposed methods work better than the three compared methods

on all environments except Swimmer. On the environment Swimmer, we still achieved a satisfactory result as expected.

Table 7.5: Relative return based sample efficiency of four methods on six environments

Environment	TRPO	ME-MF	ME-TRPO	DA-TRPO
Cart Pole	0.5000	0.6997	0.8990	**1.0000**
Pendulum	0.7594	0.5000	0.8543	**1.0000**
Inverted Pendulum	0.6812	0.5000	0.9500	**1.0000**
Swimmer	0.9285	**1.0000**	0.5000	0.8434
Half Cheetah	0.6008	0.5000	0.9855	**1.0000**
Walker2d	0.5000	0.5134	0.5120	**1.0000**
Mean	0.6616	0.6188	0.7835	**0.9739**

On the other hand, we evaluate the sample efficiency based on the relative return defined in Chapter 6 for four RL methods on six benchmark environments, as shown in Table 7.5. Except for Swimmer, our DA-TRPO has the highest sample efficiency among the four methods on the remained five environments. From the last row of Table 7.5, overall, the highest sample efficiency is from our proposed approach DA-TRPO.

In summary, these experimental results convincingly verify the effectiveness and efficiency of our proposed methods, and thus model ensemble trust region policy optimization is enhanced further.

7.6.3 Ablation Experiments

This sub-section will provide three ablation experiments to illustrate the effectiveness of some components in our proposed algorithm.

7.6.3.1 Ablation Experiment on RauNet

Our new RauNet for dynamics model consists of three modules: UNet, ResNet, and attention modules. Generally speaking, the attention could not become an independent deep network. Based on the remaining two modules, three deep networks have been built, i.e., UNet, ResNet, and their combination ResUNet, in deep learning. In this subsection, we conduct an ablation experiment on the Pendulum to validate whether our RauNet outperforms the three aforementioned deep networks, as shown in Figure 7.11.

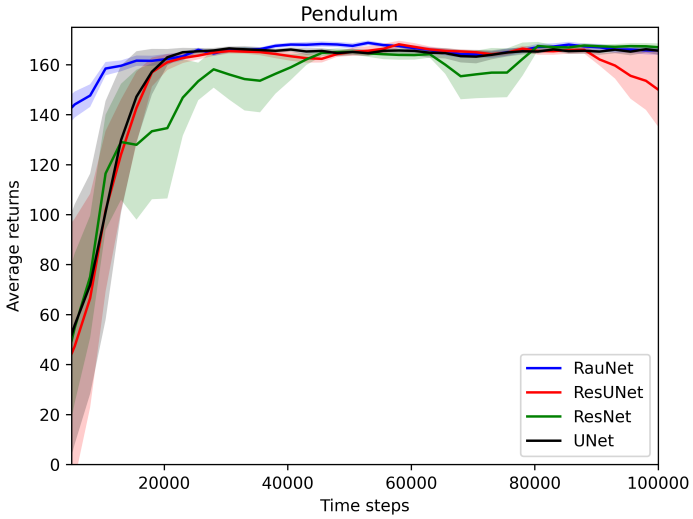


Figure 7.11: Ablation experiment on Pendulum for different network architectures.

From 7.11, we observe the following: (a) at the beginning, our RauNet achieves a higher average return than the other three networks, (b) ResNet is not stable enough, which is improved by combining UNet, and (c) After 25,000-time steps, UNet, ResUNet, and our RauNet perform almost equally. Overall, this ablation experiment demonstrates that our RauNet is more effective than its origins.

7.6.3.2 Ablation experiment on diversity-driven objective

In our study, we add the normalized Hilbert-Schmidt independence criterion as a regularization term in our diversity-driven objective. Here, we validate whether this updated objective works or not. For the Pendulum environment, the experimental results with and without NHSIC are shown in Figure 7.12. The red curve with diversity achieves a higher initial average return than that without diversity, and then go to be stable. The blue curve without diversity increases quickly before 20000 time steps, After 20000 time steps, the curve without diversity is slightly lower than and approximate to the curve with diversity. It is demonstrated that our diversity-driven objective can improve the performance of our proposed method.

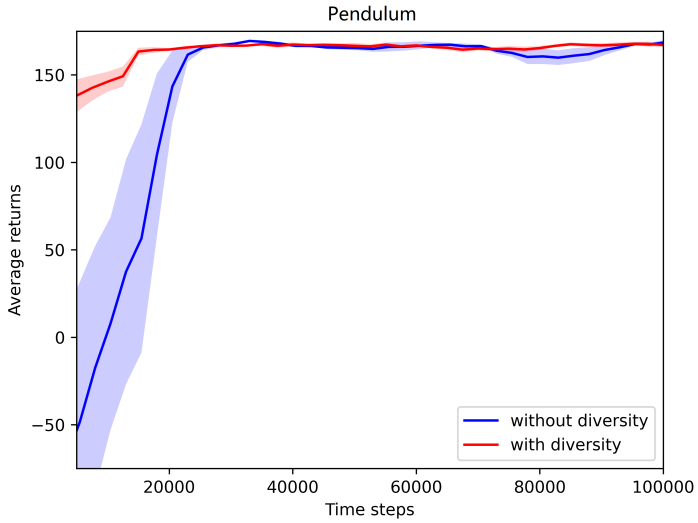


Figure 7.12: Ablation experiment on Pendulum for the diversity-driven objective.

7.6.3.3 Ablation experiment on alpha values

For Renyi alpha divergence, we conduct an ablation experiment to determine whether our adaptive procedure can work well. According to our new network architecture, four different alpha values are checked, i.e., $\alpha = 0.5$ (Hellinger distance), 1.0 (KL divergence), 1.5 and 2.0 (Chi-Square divergence), which are compared with our adaptive process, as shown in Figure 7.13. RN-TRPO and RN-aTRPOs (alpha=1.5 and 2.0) perform differently from one another. RN-aTRPO (alpha=0.5) is the best result, which is approximated by our DA-TRPO. In this case, we find that our adaptive procedure can achieve almost equal performance as the best alpha does, which validates that our design is effective.

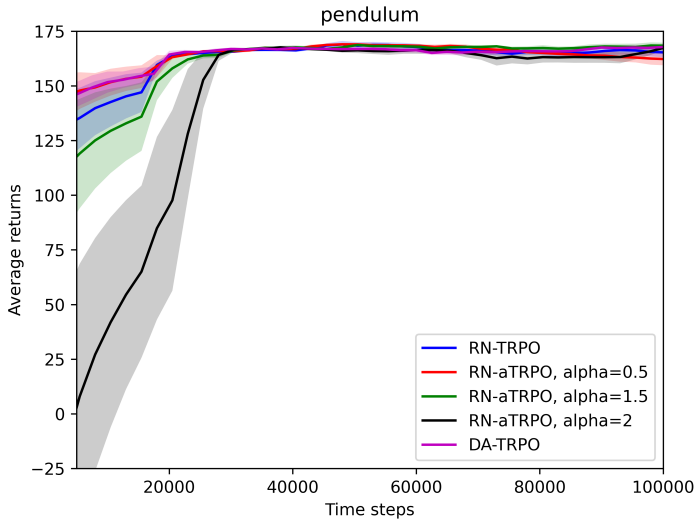


Figure 7.13: Ablation experiment on Pendulum for different alpha values.

7.7 Summary

In this chapter, we proposed a diversity-driven model ensemble adaptive trust region policy optimization, which consists of three key aspects: a residual attention U-net for environment dynamics models, a diversity-driven objective function, and an adaptive trust region policy optimization. Our novel model-ensemble trust region policy optimization is validated to be effective and efficient experimentally.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this chapter, we will conclude our research on four aspects, and further provide some future work.

8.1 Conclusions

In this thesis, there are four aspects of research that concentrate on the flexible trust region constraint for enhancing the performance of the policy optimization, and thus we conclude each of them as follows.

As we know, TRPO has been one of the most successful algorithms in the deep reinforcement learning community. However, it is frequently criticized for compressing the exploration ability of the agent learning due to its strict trust region constraint. Hence, it is desired to look for a strategy to prompt TRPO with a better exploration ability, which is usually realized by an entropy regularization. We argue about where to add this regularization term. The existing TRPO variants add a Shannon entropy regularization term to their objectives only, which is, in essence an indirect way to promote the exploration ability. In this thesis, our novel alternative aims to regularize the KL divergence constraint, to relax the trust region for enhancing the exploration ability directly. Our TRPO version is formulated to maximize the advantage function-based objective, subject to a threshold of KL divergence minus Shannon entropy, simply ERC-TRPO. Further, we provide a detailed solution at a code-level optimization, to conduct a fair experimental comparison. The comparative study, among the original TRPO, an

entropy regularized objective TRPO (ERO-TRPO) and our ERC-TRPO, demonstrates our proposed approach to be the most effective and efficient across six benchmark environments. To the best of our knowledge, this is the first work to regularize the constraint to relax the trust region from above effectively for policy optimization.

The primary TRPO is formulated as a nonlinear constrained optimization problem, which bounds the trust region from above. Additionally, it is widely solved approximately, and its success depends on two factors: search direction and step size. However, the existing solution only results in an upper bound for the step size search. In this thesis, we argue whether to bound the trust region from below and thus find a lower bound of the step size. To this end, we apply a reciprocal optimization technique and then build a reciprocal trust region policy optimization (rTRPO) by exchanging the objective and the KL divergence constraint. This procedure can yield a lower bound for the step size search. Moreover, fusing both trust region policy optimization and its reciprocal version into a framework results in twin trust region policy optimization (twinTRPO), whose approximate solution provides an upper bound and a lower bound to effectively control the step size search, to facilitate the solution to TRPO further. From experiments with nine benchmark environments and four comparison methods (TRPO, PPO, rTRPO, and our TwinTRPO), we show the superiority of our twinTRPO. To the best of our knowledge, it is the first research to bound the trust region from below for policy optimization.

Proximal policy optimization (PPO) has demonstrated its significance in the field of deep reinforcement learning via its clipping version. However, this clipping form only characterizes the trust region constraint indirectly. In this thesis, we argue how to retain its explicit trust region constraint. Therefore, the primary penalty version of PPO is reconsidered, and then two improvements are applied. Firstly, its penalty form is reformulated as a linearly combined one (combined PPO) to control the trade-off between two terms (the surrogate return and KL divergence), which also benefits the parameter tuning procedure. Secondly, its original KL divergence is replaced by a parametric alpha divergence, which measures the difference between two subsequent policies more elaborately to control the trust region effectively for different environments. These modifications and our new design of this reinforcement learning paradigm derive a novel PPO-like algorithm called alphaPPO. Via detailed experiments on six benchmark environments, the effectiveness and efficiency of our proposed alphaPPO have been verified, compared with clipping PPO and combined PPO. Our work first applies alpha divergence to fit different environments.

Model-based reinforcement learning has an outstanding potential to be more sample

efficient for some specific applications, which require building a dynamics model to characterize the reinforcement learning environments. By introducing ensemble learning, the model variance and bias can be controlled and reduced further. In this study, we investigate two key aspects: model learning and planning. On the one hand, via integrating residual network, U-net, and attention module, we build a deep residual attention U-net (RauNet), as our dynamics model network, that has fewer weights to be trained than the widely-used shallow neural networks. Further, we enforce the model diversity explicitly for our RauNet by defining a diversity-driven objective function using the normalized Hilbert-Schmidt independence criterion as a regularization term. On the other hand, to control the trust region, we propose an adaptive trust region optimization via two improvements: substituting parametric Renyi alpha divergence for KL divergence and correspondingly designing an adaptive process to adjust the alpha value during iterations, and thus allowing for adaptive trust region adjustment. In this case, a novel enhanced model ensemble trust region policy optimization based on residual attention U-net with diversity guarantee and adaptive trust region policy optimization is built, simply DA-TRPO. Some extensive experiments on six benchmark environments illustrate that our proposed methods are more effective and efficient, compared with three existing algorithms (i.e., trust region policy optimization and its previous model ensemble version, and single model reinforcement learning method). This work has two novelties: lightweight true deep dynamics models and adaptive alpha trust region policy optimization.

Sample efficiency is a frequently mentioned term in the literature, but it is always regarded as a qualitative concept. To the best of our knowledge, only in [82], a rough quantitative definition is given. To address this situation, we define two sample efficiency metrics: relative return based and AUC-based, which are normalized values in (0,1) and (0.5,10), respectively. Moreover, they are used to estimate the sample efficiency for our experimental comparison, to show that our proposed methods have a higher sample efficiency than those of the compared RL approaches.

In summary, our proposed methods enhance the trust region constraint in policy optimization in four aspects: 1) to relax the trust region from above utilizing entropy; 2) to bound the trust region from above and below; 3) to adjust the trust region by alpha divergence for different environments; 4) to measure the trust region adaptively in model ensemble reinforcement learning. And the effectiveness and efficiency of our methods are all verified by extensive experiments.

8.2 Future Work

Although four advanced policy optimization methods with flexible trust region constraints have been successfully proposed and effectively validated in this thesis, the research questions pursued in this study remain challenging and require some further investigation.

Our proposed methods enhance the performance of the TRPO method from different aspects, raising a new question of whether we can fuse both approaches into a single method. That is, we will apply reciprocal optimization technology to ERC-TRPO to yield a lower bound for the step size, to facilitate the solution for ERC-TRPO.

In alphaPPO and DA-TRPO, alpha divergence and Renyi alpha divergence could effectively improve their performance. This is because these parametric divergences can control and adjust the trust region properly. To better address the trust region, we should investigate whether and how the other parametric divergences work for ERC-TRPO and TwinTRPO. Also, we tune the alpha value via a grid search, which is time-consuming. We will embed the alpha value optimization into the PPO solution procedure, that is, the alpha value is set as a trained parameter in the actor network, to detect an optimal alpha value automatically.

In model ensemble reinforcement learning, the policy optimization methods used mainly come from on-policy TRPO and off-policy soft actor-critic (SAC), for the finite horizon episodic environments and discounted reward setting. Recently, the average rewards, which consider infinite-horizon ergodic environments, have received much attention. For example, TRPO and PPO have been extended to implement their average reward versions ATRPO [157] and APO [72]. In this case, we argue whether these average reward methods can improve the performance of model ensemble approaches.

Our aforementioned researches use some benchmark environments to verify the effectiveness and superiority of our proposed approaches. Further validations and deep applications are also needed, which will be executed on some real-world problems, for example, unmanned aerial vehicles.

Finally, we want to conduct the above future work, which will subsequently help us improve our techniques and enhance the trust region constraint further.

APPENDIX

A.1 Benchmark Environments

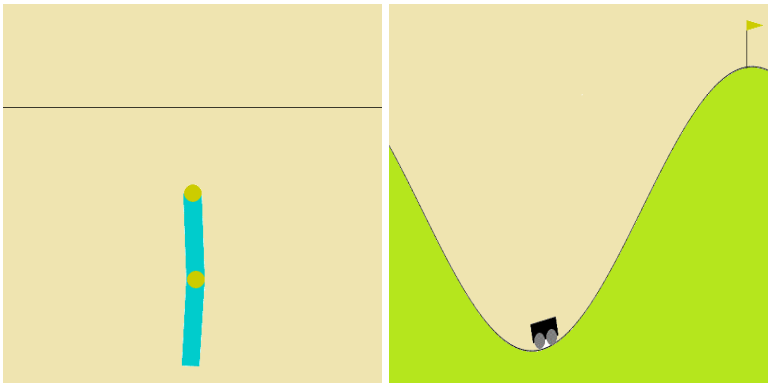
In reinforcement learning field, researchers use different benchmark environments to evaluate their reinforcement learning algorithms, most of which have been collected in the GYM website [1]. In our experiments, we choose three main kinds of environments: classic control, Box2D and MuJoCo simulator.

The classic control problem consists of five environment: Acrobot, CartPole, Mountain Car, Continuous Mountain Car, and Pendulum, as shown in Figure A.1, where two mountain car environments have the same pictorial diagram. All of these environments are stochastic in terms of their initial state, within a given range. In addition, Acrobot has noise applied to the taken action. Additionally, for both mountain car environments, the cars are underpowered to climb the mountain, so it takes some effort to reach the top.

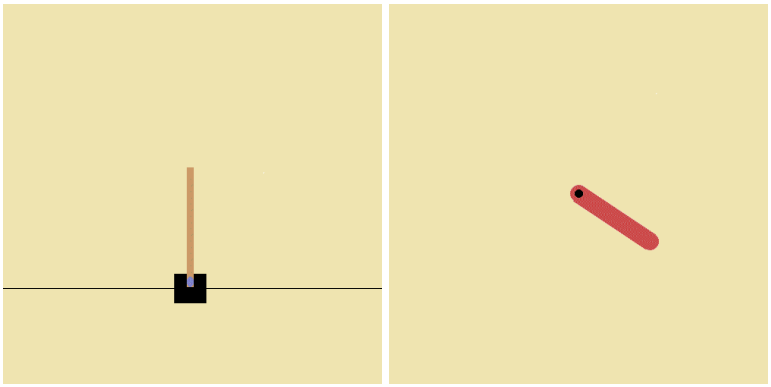
The MuJoCo indicates Multi-Joint dynamics with Contact. It is a physics engine for facilitating research and development in robotics, biomechanics, graphics and animation, and other areas, in which fast and accurate simulation is needed. In our thesis, we choose ten environments as shown in Figure A.2.

On Box2D environment, we only pick up Bipedal Walker, as shown in Figure A.3, which involves a toy game based around physics control.

[1] https://www.gymliberay.ml



(a) Acrobot (b) MountainCar

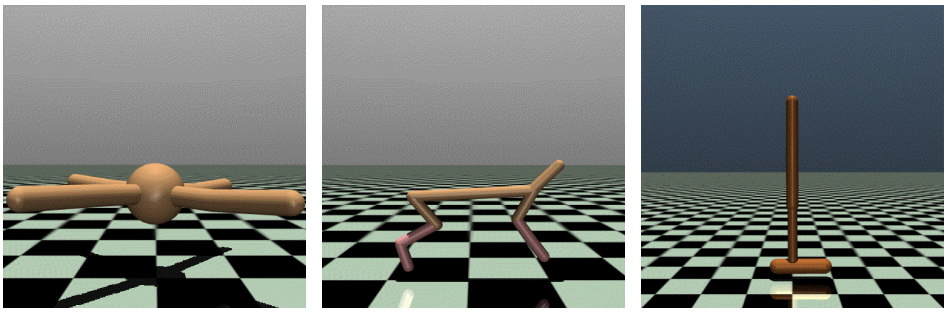


(c) Cartpole (d) Pendulum

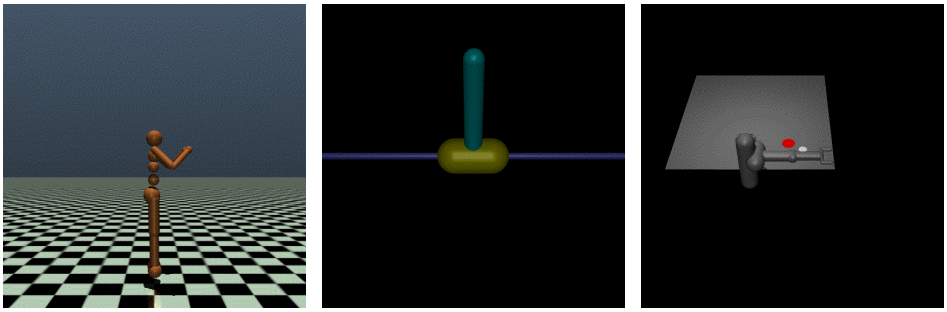
Figure A.1: Classic control environments

In Table A.1, we summarize some detailed information for these environments, including the dimensions of actio and state spaces.

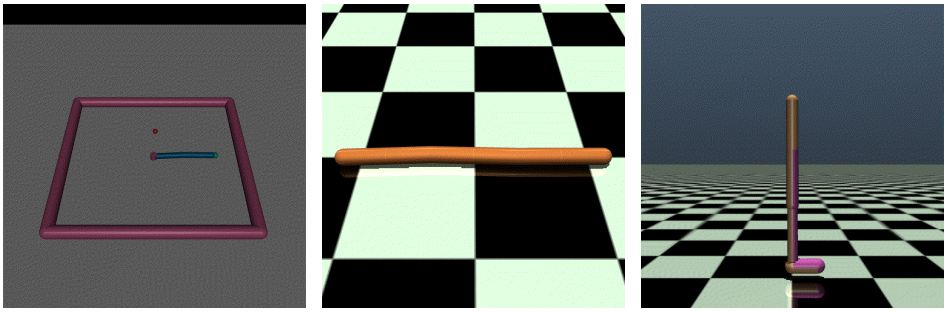
We will use some of benchmark environments to validate the performance of our proposed RL algorithms.



(a) Ant (b) Half Cheetah (c) Hopper



(d) Humanoid (e) Inverted Pendulum (f) Pusher



(g) Reacher (h) Swimmer (i) Walker2d

Figure A.2: MuJoCo simulation environments

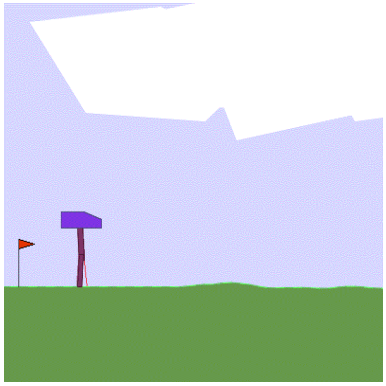


Figure A.3: Bipedal Walker from Box2D

Table A.1: Some basic information for benchmark environments

Environment	State space	Action space
Classic control		
Acrobat	6	3d
Cartpole	4	2d
Mountian	2	1
Pendulum	3	1
MuJoCo		
Ant	27	8
Humanoid	376	17
Half Cheetah	17	6
Hopper	11	3
Inverted Pendulum	4	1
Pusher	27	7
Reacher	11	2
Swimmer	8	2
Walker2D	17	6
Box2D		
Bipedal Walker	24	4d

Note: the "d" indicates the discrete actions, otherwise the continuous actions

BIBLIOGRAPHY

[1] M. M. AFSAR, T. CRUMP, AND B. FAR, *Reinforcement learning based recommender systems: A survey*, ACM Computing Surveys, 55 (2023), pp. 1–37, Artilce–145.

[2] C. C. AGGARWAL, *Neural Networks and Deep Learning: A Textbook*, Springer, New York, USA, 2018.

[3] Z. AHMED, N. L. ROUX, M. NOROUZI, AND D. SCHUURMANS, *Understanding the impact of entropy on policy optimization*, in The 36th International Conference on Machine Learning, 2019, pp. 1–10.

[4] R. AKROUR, J. PAJARINEN, G. NEUMANN, AND J. PETERS, *Projections for approximate policy iteration algorithms*, in The 36th International Conference on Machine Learning, 2019, pp. 1–12.

[5] A. Z. AL-MARRIDI, A. MOHAMED, AND A. ERBAD, *Optimized blockchain-based healthcare framework empowered by mixed multi-agent reinforcement learning*, Journal of Network and Computer Applications, 224 (2024), pp. 1–16, Article–103834.

[6] L. ALZUBAIDI, J. ZHANG, A. J. HUMAIDI, A. ALDUJAILI, Y. DUAN, O. AL-SHAMMA, J. SANTAMARIA, M. A. FADHEL, M. ALAMIDIE, AND L. FARHAN, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, Journal of Big Data, 8 (2021), pp. 1–74, Article–53.

[7] S.-I. AMARI, *Alpha-divergence is unique, belonging to both f-divergence and bregman divergence classes*, IEEE Transactions on Information Theory, 55 (2009), pp. 4925–4931.

[8] K. AZIZZADENESHELI, M. K. BERA, AND A. ANANDKUMAR, *Trust region policy optimization for POMDPs*, arXiv:1810.07900v2, (2019), pp. 1–9.

[9] D. BANK, N. KOENIGSTEIN, AND R. GIRYE, *Autoencoders*, arXiv:2003.05991v2, (2021), pp. 1–22.

[10] E. BARBIERATO AND A. GATTI, *The challenges of machine learning: a critical review*, Electronics, 13 (2024), pp. 1–13, Aricle–416.

[11] A. BECK AND M. TEBOULLE, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, Operation Research Letters, 31 (2003), pp. 167–175.

[12] R. BELLMAN, *A markovian decision process*, Journal of Mathematics and Mechanics, 6 (1957), pp. 679–684.

[13] R. BELLMAN, *Dynamics Programming*, Dover Publications, Garden City, NY, USA, 2003.

[14] B. BELOUSOV AND J. PETERS, *F-divergence constrained policy improvement*, arXiv:1801.00056v2, (2018), pp. 1–20.

[15] D. P. BERTSEKAS, *Nonlinear Programming*, Athena Scientific, Belmont, MA, USA, 3rd ed., 2016.

[16] C. M. BISHOP, *Pattern Recognition and Machine Learning*, Srpinger, New York, USA, 2005.

[17] P.-T. D. BOER, D. P. KROESE, S. MANNOR, , AND R. Y. RUBINSTEIN, *A tutorial on the cross-entropy method*, Annals of Operations Research, 134 (2005), pp. 19–67.

[18] G. BRAUWERS AND F. FRASINCAR, *A general survey on attention mechanisms in deep learning*, IEEE Transactions on Knowledge and Data Engineering, 35 (2023), pp. 3279–3298.

[19] A. M. CARRINGTON, D. G. MANUEL, P. W. FIEGUTH, T. RAMSAY, V. OSMANI, B. WERNLY, C. BENNETT, S. HAWKEN, O. MAGWOOD, Y. SHEIKH, M. MCINNES, AND A. HOLZINGER, *Deep ROC analysis and AUC as balanced average accuracy, for improved classifier selection, audit and explanation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 45 (2023), pp. 329–341.

[20] S. CAYCI, N. HE, AND R. SRIKANT, *Covergence of entropy regularized natural policy gradient methods with linear function approximation*, arXiv:2106.04096v3, (2022), pp. 1–14.

[21] S. CEN, C. CHENG, Y. CHEN, Y. WEI, AND Y. CHI, *Fast global convergence of natural policy gradient methods with entropy regularization*, Operations Research, 70 (2022), pp. 2563–2578.

[22] B. CHEN, Y. XU, AND A. SHRIVASTAVA, *Fast and accurate stochastic gradient estimation*, in The 33rd Annual Conference on Neural Information Processing Systems, 2019, pp. 12339–12349.

[23] G. CHEN, Y. PENG, AND M. ZHANG, *An adaptive clipping approach for proximal policy optimization*, arXiv: 1804.06461v1, (2018), pp. 1–8.

[24] Y. CHENG, Q. GUO, AND X. WANG, *Proximal policy optimization with advantage reuse competition*, IEEE Transactions on Artificial Intelligence, 5 (2024), pp. 3915–3925.

[25] X. CHU, *A strong on-policy competitor to PPO*, Openview.net, (2021), pp. 1–11.

[26] K. CHUA, R. CALANDRA, R. MCALLISTER, AND S. LEVINE, *Deep reinforcement learning in a handful of trials using probabilistic dynamics models*, in The 32nd Annual Conference on Neural Information Processing Systems, 2018, pp. 4759–4770.

[27] A. CICHOCKI AND S.-I. AMARI, *Families of alpha- beta- and gamma- divergence: flexible and robust measures of similarities*, Entropy, 12 (2010), pp. 1532–1568.

[28] C. CORTES, M. MOHRI, AND A. ROSTAMIZADEH, *Algorithms for learning kernels based on centered alignment*, Journal of Machine Learning Research, 13 (2012), pp. 795–828.

[29] T. M. COVER AND J. A. THOMAS, *Elemnets of Information Theory*, Wiley and Sons, New York, USA, 2nd ed., 2006.

[30] N. CRISTIANINI, J. SHAWE-TAYLOR, A. ELISSEEFF, AND J. S. KANDOLA, *On kernel-target alignment*, in The 15th Annual Conference on Neural Information Processing Systems, 2002, pp. 367–373.

[31] V.-A. DARVARIU, S. HAILES, AND M. MUSOLESI, *Graph reinforcement learning for combinatorial optimization: A survey and unifying perspective*, arXiv:2404.06492, (2024), pp. 1–24.

[32] M. DEISENROTH AND C. E. RASMUSSEN, *Pilco: A model-based and data-efficient approach to policy search*, in The 28th International Conference on Machine Learning, 2011, pp. 465–472.

[33] J. DEMSAR, *Statistical comparisons of classifiers over multiple data sets*, Journal of Machine Learning Research, 7 (2006), pp. 1–30.

[34] H. DONG, Z. DING, AND S. ZHANG, *Deep Reinforcement Learning: Fundamentals, Research and Applications*, Springer, Singapore, 2020.

[35] R. F. J. DOSSA, S. HUANG, S. ONTANON, AND T. MATSUBARA, *An empirical investigation of early stopping optimization in proximal policy optimization*, IEEE Access, 9 (2021), pp. 117981–117992.

[36] V. ELVIRA AND L. MARTINO, *Advances in importance sampling*, arXiv:2102.05407v3, (2022), pp. 1–21.

[37] L. ENGSTROM, A. ILYAS, S. SANTURKAR, D. TSIPRAS, F. JANOOS, L. RUDOLPH, AND A. MADRY, *Implementation matters in deep policy gradients: A case study on PPO and TRPO*, in The 7th International Conference on Learning Representations, 2020, pp. 1–14.

[38] B. EYSENBACK AND S. LEVINE, *Maximum entropy RL (provably) solves some robust RL problems*, in The 13rd International Conference on Learning Representations, 2022, pp. 1–12.

[39] P. FAVATI AND M. PAPPALARDO, *On the reciprocal vector optimization problems*, Journal of Optimization Theory and Applications, 47 (1985), pp. 181–193.

[40] T. FAWCETT, *An introduction to ROC analysis*, Pattern Recognition Letters, 27 (2006), pp. 861–874.

[41] G. FRANCESCHELLI AND M. MUSOLESI, *Reinforcement learning for generative AI: State of the art, opportunities and open research challenges*, Journal of Artificial Intelligence Research, 79 (2024), pp. 417–446.

[42] G. H. GOLUB AND C. F. V. LOAD, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[43] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep Learning*, MIT press, Cambridge MA, USA, 2016.

[44] A. GRETTON, O. BOUSQUET, AND A. S. AMD BERNHARD SCHOLKOPF, *Measuring statistical dependence with Hilert-Schmidt norm*, in The 16th International Conference on Algorithmic Learning Theory, 2005, pp. 63–77.

[45] I. GRONDMAN, L. BUSONIU, G. A. D. LPES, AND R. BABUSKA, *A survey of actor-critic reinforcement learning: Standard and natural policy gradients*, IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 42 (2012), pp. 1291–1307.

[46] Y. GU, Y. CHENG, C. L. P. CHEN, AND X. WANG, *Proximal policy optimization with policy feedback*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52 (2022), pp. 4600–4610.

[47] Y. GUO, H. LONG, X. DUAN, K. FENG, M. LI, AND X. MA, *CIM-PPO: proximal policy optimization with liu-correntropy induced metric*, arXiv: 2110.10522v2, (2022), pp. 1–12.

[48] T. HAARNOJA, A. ZHOU, P. ABBEEL, AND S. LEVINE, *Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor*, in The 35th International Conference on Machine Learning, 2018, pp. 1–10.

[49] P. HARREMOES, *Interpretations of Renyi entropies and divergences*, Physica A: Statistical Mechanics and its Applications, 365 (2006), pp. 57–62.

[50] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in The 29th IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[51] M. HU, J. ZHANG, L. MATKOVIC, T. LIU, AND X. YANG, *Reinforcement learning in medical image analysis: Concepts, applications, challenges, and future directions*, Journal of Applied Clinical Medical Physics, 24 (2023), pp. 1–21, Article–13898.

[52] F. HUANG, S. GAO, AND H. HUANG, *Bregman gradient policy optimization*, in The 6th International Conference on Learning Representations, 2022, pp. 1–22.

[53] M. JANNER, J. FU, M. ZHANG, AND S. LEVINE, *When to trust your model: Model-based policy optimization*, in The 33rd Conference on Neural Information Processing Systems, 2019, pp. 12498–12509.

[54] D. K. JHA, A. U. RAGHUNATHAN, AND D. ROMERES, *Quasi-Newton trust region policy optimization*, in The 3rd Conference on Robot Learning, 2019, pp. 1–10.

[55] S. KAKADE AND J. LANGFORD, *Approximately optimal approximate reinforcement learning*, in The 18th International Conference on Machine Learning, vol. 2, 2002, pp. 267–274.

[56] D. P. KINGMA AND J. L. BA, *ADAM: A method for stochastic optimization*, in The 3rd International Conference on Learning Representations, 2015, pp. 1–15.

[57] B. R. KIRAN, I. SOBH, V. TALPAERT, P. MANNION, A. A. A. SALLAB, S. YOGAMANI, AND P. PEREZ, *Deep reinforcement learning for autonomous driving: A survey*, IEEE Transactions on Intelligent Transportation Systems, 23 (2022), pp. 4909–4926.

[58] T. KOBAYASHI, *Proximal policy optimization with relative Pearson divergence*, in 2021 IEEE International Conference on Robotics and Automation, 2021, pp. 8416–8421.

[59] I. KOBYZEV, S. J. D. PRINCE, AND M. A. BRUBAKER, *Normalizing flows: an introduction and review of current methods*, IEEE Transactions on Pattern Recognition and Machine Intelligence, 43 (2021), pp. 3964–3979.

[60] V. R. KONDA AND J. N. TSITSIKLIS, *Actor-critic algorithms*, in The 14th Annual Conference on Neural Information Processing Systems, 2000, pp. 1008–1014.

[61] S. KORNBLITH, M. NOROUZI, H. LEE, AND G. HINTON, *Similarity of neural network representation revisited*, in The 36th International Conference on Machine Learning, 2019, pp. 3519–3529.

[62] S. KULLBACK AND R. LEIBLER, *On information and sufficiency*, The Annals of Mathematical Statistics, 22 (1951), pp. 79–86.

[63] T. KURUTACH, I. CLAVERA, Y. DUAN, A. TAMAR, AND P. ABBEEL, *Model-ensemble trust-region policy optimization*, in The 6th International Conference on Learning Representations, 2018, pp. 1–15.

[64] P. LADOSZ, L. WENG, M. KIM, AND H. OH, *Exploration in deep reinforcement learning: A survey*, Information Fusion, 85 (2022), pp. 1–22.

[65] H. LI AND H. HE, *Multiagent trust region policy optimization*, IEEE Transactions on Neural Networks and Learning Systems, 35 (2024), pp. 12873–12887.

[66] Y. LIN, Y. LIU, F. LIN, L. ZOU, P. WU, W. ZENG, H. CHEN, AND C. MIAO, *A survey on reinforcement learning for recommender systems*, IEEE Transactions on Neural Networks and Learning Systems, 35 (2024), pp. 13164–13184.

[67] B. LIU, Q. CAI, Z. YANG, AND Z. WANG, *Neural proximal/trust region policy optimization attains globally optimal policy*, in The 33rd Annual Conference on Neural Information Processing Systems, 2019, pp. 1–12.

[68] W. LIU, P. P. POKHAREL, AND J. C. PRINCIPE, *Correntropy: properties and applications in non-gaussian signal processing*, IEEE Transactions on Signal Processing, 55 (2007), pp. 5286–5298.

[69] F.-M. LUO, T. XU, H. LAI, X.-H. CHEN, W. ZHANG, AND Y. YU, *A survey on model-based reinforcement learning*, Science China Information Sciences, 67 (2024), pp. 1–26, Article–121101.

[70] Y. LUO, H. XU, Y. LI, Y. TIAN, T. DARRELL, AND T. MA, *Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees*, in The 7th International Conference on Learning Representations, 2019, pp. 1–27.

[71] J. MA, *Entrpoy augmented reinforcement learning*, arXiv.2208.09322v1, (2022), pp. 1–9.

[72] X. MA, X. TANG, L. XIA, J. YANG, AND Q. ZHAO, *Average-reward reinforcement learning with trust region methods*, in The 30th International Joint Conference on Artificial Intelligence, 2021, pp. 2797–2803.

[73] J. MARTENS AND R. GROSSE, *Optimizing neural networks with Kronecker-factored approximate curvature*, in The 32nd International Conference on Machine Learning, 2015, pp. 1–10.

[74] M. A. MASOOD AND F. DOSHI-VELEZ, *Diversity-inducing policy gradient: Using maximum mean discrepancy to find a set of diverse policies*, arXiv:1906.00088, (2019), pp. 1–10.

[75] N. MAZYAVKINA, S. SVIRIDOV, S. IVANOV, AND E. BURNAEV, *Reinforcement learning for combinatorial optimization: A survey*, Computers and Operations Research, 134 (2021), pp. 1–15, Article–105400.

[76] J. MELBOURNE, *Strongly convex divergences*, Entropy, 22 (2020), pp. 1–20, Article–1327.

[77] J. MENDES-MOREIRA, C. SOARES, A. M. JORGE, AND J. F. D. SOUSA, *Ensemble approaches for regression: A survey*, ACM Computing Surveys, 45 (2012), pp. 1–40, Article–10.

[78] W. MENG, Q. ZHENG, Y. SHI, AND G. PAN, *An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning*, IEEE Transactions on Neural Networks and Learning Systems, 33 (2022), pp. 2223–2235.

[79] S. MILANI, N. TOPIN, M. VELOSO, AND F. FANG, *Explainable reinforcement learning: A survey and comparative review*, ACM Computing Surveys, 56 (2024), pp. 1–36, Article–168.

[80] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. A. RUSU, J. VENESS, M. G. BELLEMARE, A. GRAVES, M. A. RIEDMILLER, A. FIDJELAND, G. OSTROVSKI, S. PETERSEN, C. BEATTIE, A. SADIK, I. ANTONOGLOU, H. KING, D. KUMARAN, D. WIERSTRA, S. LEGG, AND D. HASSABIS, *Human-level control through deep reinforcement learning*, Nature, 518 (2015), pp. 529–533.

[81] T. M. MOERLAND, J. BROEKENS, A. PLAAT, AND C. M. JONKER, *Model-based reinforcement learning: A survey*, Foundations and Trends in Machine Learning, 16 (2023), pp. 1–118.

[82] S. MOHANTY, J. POONGANAM, A. GAIDON, A. KOLOBOV, B. WULFE, D. CHAKRABORTY, G. SEMETULSKIS, J. SCHAPKE, J. KUBILIUS, J. PASUKONIS, L. KLIMAS, M. HAUSKNECHT, P. MACALPINE, Q. N. TRAN, T. TUMIEL, X. TANG, X. CHEN, C. HESSE, J. HILTON, W. H. GUSS, S. GENC, J. SCHULMAN, AND K. COBBE, *Measuring sample efficiency and generalization in reinforcement learning benchmarks: Neurips 2020 procgen benchmark*, arXiv:2103.15332v1, (2021), pp. 1–27.

[83] R. MORADI, R. BERANGI, AND B. MINAEI, *A survey of regularization strategies for deep models*, Artificial Intelligence Review, 53 (2020), pp. 3947–3986.

[84] O. NACHUM, M. NOROUZI, K. XU, AND D. SCHUURMANS, *Bridging the gap between value and policy based reinforcement learning*, in The 31st Annual Conference on Neural Information Processing Systems, 2017, pp. 2772–2782.

[85] O. NACHUM, M. NOROUZI, K. XU, AND D. SCHUURMANS, *TRUST-PCL: an off-policy trust region method for continuous control*, in The 6th International Conference on Learning Representations, 2018, pp. 1–14.

[86] A. NAGABANDI, G. KAHN, R. S. FEARING, AND S. LEVINE, *Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning*, arXiv:1708.02596v2, (2017), pp. 1–10.

[87] N. J. NILSSON, *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann Publishers, San Mateo, CA, USA, 1998.

[88] Z. NIU, G. ZHONG, AND H. YU, *A review on the attention mechanism of deep learning*, Neurocomputing, 452 (2021), pp. 48–62.

[89] F. OTTO, P. BECKER, N. A. VIEN, H. C. ZIESCHE, AND G. NEUMANN, *Differentible trust region layers for deep reinforcement learning*, in The 9th International Conference on Learning Representations, 2021, pp. 1–12.

[90] F. PAN, J. HE, D. TU, AND Q. HE, *Trust the model when it is confident: Masked model-based actor-critic*, in The 34th Annual Conference on Neural Information Processing Systems, 2020, pp. 10537–10546.

[91] A. PLAAT, *Deep Reinforcement Learning*, Springer, Singapore, 2022.

[92] A. PLAAT, W. KOSTERS, AND M. PREUSS, *High-accuracy model-based reinforcement learning, a survey*, Artificial Intelligence Reviews, 56 (2023), pp. 9541 – 9573.

[93] R. F. PRUDENCIO, M. R. O. A. MAXIMO, AND E. L. COL, *A survey on offline reinforcement learning: Taxonomy, review, and open problems*, IEEE Transactions on Neural Networks and Learning Systems, 35 (2024), pp. 10237–10257.

[94] M. L. PUTERMAN, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley and Sons, Hoboken, NJ, USA, 2005.

[95] J. QUEENEY, I. C. PASCHALIDIS, AND C. CASSANDRAS, *Generalized proximal policy optimization with sample reuse*, in The 35th Annual Conference on Neural Information Processing Systems, 2021, pp. 1–11.

[96] J. QUEENEY, I. C. PASCHALIDIS, AND C. G. GASSANDRAS, *Uncertainty-aware policy optimization: a robust, adaptive trust region approach*, in The 35th AAAI Conference in Artificial Intelligence, 2021, pp. 9377–9385.

[97] F. RAIBER AND O. KURLAND, *Kullback-leibler divergence revisited*, in The 2017 ACM SIGIR International Conference on the Theory of Information Retrieval, 2017, pp. 117–124.

[98] A. V. RAO, *A survey of numerical methods for optimal control*, Advances in the Astronautical Sciences, 135 (2009), pp. 497–528.

[99] Y. REN, L. ZHANG, AND P. N. SUGANTHAN, *Ensemble classification and regression - recent developments, applications and future directions*, IEEE Computational Intelligence Magazine, 11 (2016), pp. 41–53.

[100] A. RENYI, *On measures of entropy and information*, in The 4th Berkeley Sympon on Mathematical Statist and Probability, 1961, pp. 547–561.

[101] H. ROJAS AND A. LOGACHOV, *Central limit theorem on symmetric Kullback-Leibler (KL) divergence*, arXIv:2401.16524v1, (2024), pp. 1–14.

[102] O. RONNEBERGER, P. FISCHER, AND T. BROX, *U-net: Convolutional networks for biomedical image segmentation*, in The 2015 International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241.

[103] S. ROOSRAIE AND M. M. EBADZADEH, *EnTRPO: trust region policy optimization method with entropy regularization*, arXiv:2110.13373, (2021), pp. 1–11.

[104] S. RUSSELL AND P. NORVIG, *Artificial Intelligence: A Modern Approach*, Pearson Press, Old Tappan, NJ, USA, 4th ed., 1998.

[105] I. H. SARKER, *Machine learning: Algorithms, real-world applications and research directions*, SN Computer Science, 2 (2021), pp. 1–21, Artilce–160.

[106] J. SCHULMAN, S. LEVINE, P. MORITZ, M. JORDAN, AND P. ABBEEL, *Trust region policy optimization*, in The 32nd International Conference on Machine Learning, 2015, pp. 1889–1897.

[107] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, AND O. KLIMOV, *Proximal policy optimization algorithms*, arXiv:1707.06347v2, (2017), pp. 1–12.

[108] M. SHAFIQ AND Z. GU, *Deep residual learning for image recognition: A survey*, Applied Sciences, 12 (2022), pp. 1–43, Article–8972.

[109] L. SHANI, Y. EFRONI, AND S. MANNOR, *Adaptive trust region policy optimization: global convergence and faster rates for regularized mdps*, in The 34th AAAI Conference on Artificial Intelligence, 2020, pp. 5668–5675.

[110] L. SHARMA AND P. K. GARG, *Artificial Intelligence: Technologies, Applications, and Challenges*, Chapman and Hall/CRC Press, Boca Raton, FL, USA, 2021.

[111] N. SIDDIQUE, S. PAHEDING, C. P. ELKIN, AND V. DEVABHAKTUNI, *U-net and its variants for medical image segmentation: A review of theory and applications*, IEEE Access, 9 (2021), pp. 82031–82057.

[112] D. SILVER, A. HUANG, C. J. MADDISON, A. GUEZ, L. SIFRE, G. D. DRIESSCHE, J. SCHRITTWIESER, I. ANTONOGLOU, V. PANNEERSHELVAM, M. LANCTOT, S. DIELEMAN, D. GREWE, J. NHAM, N. KALCHBRENNER, I. SUTSKEVER, T. LILLICRAP, M. LEACH, K. KAVUKCUOGLU, T. GRAEPEL, AND D. HASSABIS, *Mastering the game of go with deep neural networks and tree search*, Nature, 529 (2016), pp. 484–489.

[113] D. SILVER, J. SCHRITTWIESER, K. SIMONYAN, I. ANTONOGLOU, A. HUANG, A. GUEZ, T. HUBERT, L. BAKER, M. LAI, A. BOLTON, Y. T. CHEN, T. LILLICRAP, F. HUI, L. SIFREAND, G. V. D. DRIESSCHE, T. GRAEPEL, AND D. HASSABIS, *Mastering the game of GO without human knowledge.*, Nature, 550 (2017), pp. 354–359.

[114] S. SON, L. Y. ZHENG, R. SULLIVAN, Y.-L. QIAN, AND M. C. LIN, *Gradient informed proximal policy optimization*, in The 37th Annual Conference on Neural Information Processing Systems, 2023, pp. 1–12.

[115] Y. SONG, P. N. SUGANTHAN, W. PEDRYCZ, J. OU, Y. HE, Y. CHEN, AND Y. WU, *Ensemble reinforcement learning: A review*, arXiv:2303.02618v3, (2023), pp. 1–34.

[116] K. SOUCHLERIS, G. K. SIDIROPOULOS, AND G. A. PAPAKOSTAS, *Reinforcement learning in game industry - review, prospects and challenges*, Applied Sciences, 13 (2023), pp. 1–23, Article–2443.

[117] E. W. STEYERBERG, A. J. VICKERS, N. R. COOK, T. GERDS, M. GONEN, N. OBUCHOWSKI, M. J. PENCINA, , AND M. W. KATTANE, *Assessing the performance of prediction models: A framework for some traditional and novel measures*, Epidemiology, 21 (2009), pp. 128–138.

[118] M. SUGIYAMA, *Introduction to Machine Learning*, Elsevier, New York, USA, 2016.

[119] M. SUN, V. KURIN, G. LIU, S. DELIN, T. QIN, K. HOFMANN, AND S. WHITESON, *You may not need ratio clipping in PPO*, arXiv:2022.00079v1, (2022), pp. 1–20.

[120] R. K. SUNDARAM, *A First Course in Optimization Theory*, Cambridge University Press, Cambridge, UK, 1996.

[121] R. SUTTON, *Dyna, an integrated architecture for learning, planning, and reacting*, ACM SIGART Bulletin, 2 (1991), pp. 160–163.

[122] R. SUTTON, *Planning by incremental dynamic programming*, in The 9th International Conference on Machine Learning, 1991, pp. 353–357.

[123] R. SUTTON AND A. BARTO, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge MA, USA, 2nd ed., 2018.

[124] R. SUTTON/VAPCE0MM, *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming*, in The 7th International Conference on Machine Learning, 1990, pp. 216–224.

[125] Y. TANG AND S. AGRAWAL, *Boosting trust region policy optimization with normalizing flows policy*, arXiv:1809.10326v3, (2019), pp. 1–14.

[126] A. TERPIN, N. LANZETTI, B. YARDIM, F. DORFLER, AND G. RAMPONI, *Trust region policy optimization with optimal transport discrepancies: Duality and algorithm for continuous actions*, in The 36th Annual Conference on Neural Information Processing Systems, 2022, pp. 1–12.

[127] Y. TIAN AND Y. ZHANG, *A comprehensive survey on regularization strategies in machine learning*, Information Fusion, 80 (2022), pp. 146–166.

[128] S. T. TOKDAR AND R. E. KASS, *Importance sampling: a review*, WIREs Computational Statistics, 2 (2010), pp. 54–60.

[129] M. TSCHANNEN, O. BACHEM, AND M. LUCIC, *Recent advances in autoencoder-based representation learning*, in The 3rd Workshop on Bayesian Deep Learning in NeurIPS 2018, 2018, pp. 1–25.

[130] T. VAN ERVEN AND P. HARREMOES, *Renyi divergence and Kullback-Leibler divergence*, IEEE Transactions on Information Theory, 60 (2014), pp. 3797–3820.

[131] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. KAISER, AND I. POLOSUKHIN, *Attention is all you need*, in The 31st Annual Conference on Neural Information Processing Systems, 2017, pp. 5998–6008.

[132] Q. WANG, Y. LI, AND J. XIONG, *Divergence-augemented policy optimization*, in The 33rd Annual Conference on Neural Information Processing Systems, 2019, pp. 1–11.

[133] T. WANG, X. BAO, I. CLAVERA, J. HOANG, Y. WEN, E. LANGLOIS1, S. ZHANG, G. ZHANG, P. ABBEEL, AND J. BA, *Benchmarking model-based reinforcement learning*, arXiv:1907.02057v1, (2019), pp. 1–25.

[134] X. WANG, S. WANG, X. LIANG, D. ZHAO, J. HUANG, X. XU, B. DAI, AND Q. MIAO, *Deep reinforcement learning: A survey*, IEEE Transactions on Neural Networks and Learning Systems, 35 (2024), pp. 5064–5078.

[135] Y. WANG, H. HE, AND X. TAN, *Truly proximal policy optimization*, in The 35th Uncertainty in Artificial Intelligence Conference, 2019, pp. 113–122.

[136] Y. WANG, H. HE, X. TAN, AND Y. GAN, *Trust region-guided proximal policy optimization*, in The 33rd Annual Conference on Neural Information Processing Systems, 2019, pp. 626–636.

[137] Z. WANG, J. WANG, Q. ZHOU, B. LI, AND H. LI, *Sample-efficient reinforcement learning via conservative model-based actor-critc*, in The 36th AAAI Conference on Artificial Intelligence, 2022, pp. 8612–8620.

[138] C. J. C. H. WATKINS AND P. DAYAN, *Q-learning*, Machine Learning, 8 (1992), pp. 279–292.

[139] J. WATT, *Machine Learning Refined*, Cambridge University Press, Cambridge, UK, 2020.

[140] D. J. WHITE, *A survey of applications of markov decision processes*, Journal of the Operational Research Society, 44 (1993), pp. 1073–1096.

[141] R. J. WILLIAMS, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, Machine Learning, 8 (1992), pp. 229–256.

[142] T. WU, B. ZHU, R. ZHANG, Z. WEN, K. RAMCHANDRAN, AND J. JIAO, *Pairwise proximal policy optimization: Harnessing relative feedback for LLM alignment*, ArXiv:2310.00212v3, (2023), pp. 1–19.

[143] Y. WU, E. MANSIMOV, S. LIAO, R. GROSSE, AND J. BA, *Scalable trust-region method for deep reinforcement learning using Kronecker-factored approxiamtion*, in The 31st Annual Conference on Neural Information Processing Systems, 2017, pp. 1–10.

[144] G. XIE, W. ZHANG, Z. HE, AND G. LI, *Upper confident bound advantage function proximal policy optimization*, Cluster Computing, 26 (2023), pp. 2001–2010.

[145] Z. XIE, Q. ZHANG, AND R. XU, *Simple policy optimization*, arXiv:2401.16025v6, (2024), pp. 1–25.

[146] M. YAMADA, T. SUZUKI, T. KANAMORI, H. HACHIYA, AND M. SUGUYAMA, *Relative density-ratio estimation for robust distribution comparison*, Neural Computation, 25 (2013), pp. 1324–1370.

[147] M. YAMADA, J. TANG, J. LUGO-MARTINEZ, E. HODZIC, R. SHRESTHA, A. SAHA, H. OUYANG, D. YIN, H. MAMITSUKA, C. SAHINALP, P. RADIVOJAC, F. MENCZER, AND Y. CHANG, *Ultra high-dimensional nonlinear feature selection for big biological data*, IEEE Transactions on Knowledge and Data Engineering, 30 (2018), pp. 1041–4347.

[148] L. YANG, Y. ZHANG, G. ZHENG, Q. ZHENG, P. LI, J. HUANG, AND G. PAN, *Policy optimization with stochastic mirror descent*, in The 36th AAAI Conference on Artificial Intelligence, 2022, pp. 8823–8831.

[149] Y. YANG, H. MODARES, K. G. VAMVOUDAKIS, AND F. L. LEWIS, *Cooperative finitely excited learning for dynamical games*, IEEE Transactions on Cybernetics, 54 (2023), pp. 797–810.

[150] Z. YANG, H. QU, M. FU, W. HU, AND Y. ZHAO, *A maximum divergence approach to optimal policy in deep reinforcement learning*, IEEE Transactions on Cybernetics, 53 (2023), pp. 1499–1510.

[151] Q. YIN, T. YU, S. SHEN, J. YANG, M. ZHAO, W. NI, K. HUANG, B. LIANG, AND L. WANG, *Distributed deep reinforcement learning: A survey and a multi-player multi-agent learning toolbox*, Machine Intelligence Research, 21 (2024), pp. 411–310.

[152] S. YU, L. S. GIRALDO, AND J. PRINCIPE, *Information-theoretic methods in deep neural networks: recent advances and emerging opportunities*, in The 30th International Joint Conference on Artificial Intelligence, 2021, pp. 4669–4678.

[153] Y. YU, *Towards sample efficient reinforcement learning*, in The 27th International Joint Conference on Artificial Intelligence, 2018, pp. 5739–5743.

[154] H. ZHANG, S. BAI, X. LAN, D. HSU, AND N. ZHENG, *Hindsight trust region policy optimization*, arXiv:1907.12439v5, (2021), pp. 1–7.

[155] Q. ZHANG, S. LENG, X. MA, Q. LIU, X. WANG, B. LIANG, Y. LIU, AND J. YANG, *Cvar-constrained policy optimization for safe reinforcement learning*, IEEE Transactions on Neural Networks and Learning Systems, (2023), pp. 1–12, Early Access, 10.1109/TNNLS.2023.3331304.

[156] Y. ZHANG, J. PAN, L. K. LI, W. LIU, Z. CHEN, X. LIU, AND J. WANG, *On the properties of Kullback-Leibler divergence between multivariate gaussian distributions*, in The 37th Annual Conference on Neural Information Processing Systems, 2023, pp. 58152–58165.

[157] Y. ZHANG AND K. W. ROSS, *On-policy deep reinforcement learning for the average-reward criterion*, in The 38th International Conference on Machine Learning, 2021, pp. 1–11.

[158] Z. ZHANG, Q. LIU, Y. LI, K. LIN, AND L. LI, *Safe reinforcement learning in autonomous driving with epistemic uncertainty estimation*, IEEE Transac-

tions on Intelligent Transportation Systems, (2024), pp. 1–14, Early Access, 10.1109/TITS.2024.3397700.

[159] H. ZHAO, W. TANG, AND D. D. YAO, *Policy optimization for continuous reinforcement learning*, in The 37th Annual Conference on Neural Information Processing Systems, 2023, pp. 1–13.

[160] M. ZHAO, Y. LI, AND Z. WEN, *A stochastic trust-region framework for policy optimization*, arXiv:1911.11640v1, (2019), pp. 1–26.

[161] W. ZHAO, H. JIANG, AND J. XIE, *Fast proximal policy optimization*, in The 6th Asian Conference on Pattern Recognition, 2021, pp. 73–86.

[162] S. K. ZHOU, H. N. LE, K. LUU, H. V NGUYEN, AND N. AYACHE, *Deep reinforcement learning in medical imaging: A literature review*, Medical Image Analysis, 73 (2021), pp. 1–38, Article–102193.

[163] W. ZHU AND A. ROSENDO, *A functional clipping approach for policy optimization algorithm*, IEEE Access, 9 (2021), pp. 96056–96063.